Optimal Permutation Routing on Hexagonal Networks

Ignasi Sau

Mascotte project, CNRS-INRIA-UNSA, France

Janez Žerovnik

IMFM and University of Maribor, Slovenija

COST 293

Outline

- Introduction
  - Statement of the problem
  - Topologies
  - Preliminaries
  - Example
- Algorithm
  - Description
  - Correctness
  - Optimality
- Conclusions

Permutation routing

- The **permutation routing** problem is a **packet routing** problem.
- Each processor is the **origin of at most one packet** and the **destination of no more than one packet**.
- The goal is to **minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \rightarrow S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \rightarrow S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

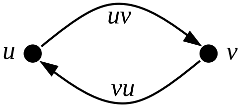
- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \rightarrow S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal:** **minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \to S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Assumptions

- We consider the **store-and-forward** and **Δ-port** model.
- **Full duplex link**: packets can be sent in the two directions of the link **simultaneously**.

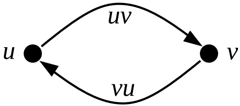


- We focus on **full-duplex hexagonal networks**.
- If the network is **half-duplex** $\rightarrow$

  2 factor approximation algorithm from an optimal algorithm for the full-duplex case, by introducing *odd-even* steps.

Assumptions

- We consider the **store-and-forward** and **Δ-port** model.
- **Full duplex link**: packets can be sent in the two directions of the link **simultaneously**.



- We focus on **full-duplex hexagonal networks**.
- If the network is **half-duplex** $\longrightarrow$

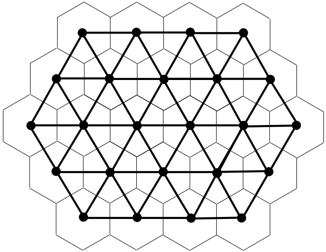
  2 factor approximation algorithm from an optimal algorithm for the full-duplex case, by introducing *odd-even* steps.

Network topologies

- There is an **ambiguity** in the notation in the literature:

$$\text{triangular grid} \leftrightarrow \text{hexagonal network},$$
$$\text{hexagonal grid} \leftrightarrow \text{honeycomb network}.$$

- Hexagonal network (△) and hexagonal tessellation (⬡):



- Hexagonal networks are finite subgraphs of the triangular grid.

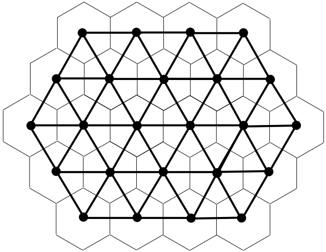
- In this work we study ***convex* hexagonal networks**.

Network topologies

- There is an **ambiguity** in the notation in the literature:

$$\text{triangular grid} \leftrightarrow \text{hexagonal network},$$
$$\text{hexagonal grid} \leftrightarrow \text{honeycomb network}.$$

- Hexagonal network (△) and hexagonal tessellation (⬡):



- Hexagonal networks are finite subgraphs of the triangular grid.
- In this work we study ***convex* hexagonal networks**.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms:

- 2-circulant graphs, square grids.
- Hexagonal networks: **Two-terminal routing**
  (only one message to be sent)

-In this work we find an **optimal permutation routing algorithm for hexagonal full-duplex networks**.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms:

- 2-circulant graphs, square grids.
- Hexagonal networks: **Two-terminal routing** (only one message to be sent)

-In this work we find an **optimal permutation routing algorithm for hexagonal full-duplex networks**.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms:

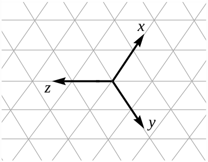
- 2-circulant graphs, square grids.
- Hexagonal networks: **Two-terminal routing**
  (only one message to be sent)

-In this work we find an **optimal permutation routing algorithm for hexagonal full-duplex networks**.

Notation and preliminary results

Nocetti, Stojmenović and Zhang [IEEE TPDS'02]:

Representation of the relative address of the nodes on a generating system ***i***, ***j***, ***k*** on the directions of the three axis x, y, z.



- This address **is not unique**, but we have that, being (a, b, c) and (a', b', c') the addresses of two $D - S$ pairs,

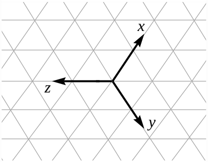
$$(a, b, c) = (a', b', c') \quad \Leftrightarrow \quad \exists \text{ an integer } d \text{ such that}$$

$$a' = a + d,$$
$$b' = b + d,$$
$$c' = c + d.$$

Notation and preliminary results

Nocetti, Stojmenović and Zhang [IEEE TPDS'02]:

Representation of the relative address of the nodes on a generating system ***i***, ***j***, ***k*** on the directions of the three axis x, y, z.



- This address **is not unique**, but we have that, being (a, b, c) and (a', b', c') the addresses of two $D - S$ pairs,

$$(a, b, c) = (a', b', c') \quad \Leftrightarrow \quad \exists \text{ an integer } d \text{ such that}$$

$$a' = a + d,$$
$$b' = b + d,$$
$$c' = c + d.$$

Notation and preliminary results (2)

- A relative address $D - S = (a, b, c)$ is of the ***shortest path form*** if
  - there is a path C from S to D, C=$a\mathbf{i}$+$b\mathbf{j}$+$c\mathbf{k}$,
  - and C has the shortest length over all paths going from S to D.

Theorem (*NSZ'02*)

*An address (a, b, c) is of the **shortest path form** if and only if*

i) ***at least one component is zero*** *(that is, $abc = 0$),*

ii) *and **any two components do not have the same sign** (that is, $ab \leq 0,\ ac \leq 0,$ and $bc \leq 0$).*

Notation and preliminary results (2)

- A relative address $D - S = (a, b, c)$ is of the ***shortest path form*** if
  - there is a path C from S to D, $C = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$,
  - and C has the shortest length over all paths going from S to D.

Theorem (*NSZ'02*)

*An address (a, b, c) is of the **shortest path form** if and only if*

i) ***at least one component is zero*** *(that is, $abc = 0$),*

ii) *and **any two components do not have the same sign***
(that is, $ab \leq 0,\ ac \leq 0,$ and $bc \leq 0$).

Notation and preliminary results (3)

Corollary (*MSZ'02*)

*Any address has a **unique** shortest path form.*

Corollary (*MSZ'02*)

If $D - S = (a, b, c)$, then the shortest path form is one of those:

$$(0, b-a, c-a),$$
$$(a-b, 0, c-b),$$
$$(a-c, b-c, 0),$$

and thus:

$$|D - S| = \min(|b-a| + |c-a|, |a-b| + |c-b|, |a-c| + |b-c|).$$

Notation and preliminary results (3)

Corollary (*MSZ'02*)

*Any address has a **unique** shortest path form.*

Corollary (*MSZ'02*)

If $D - S = (a, b, c)$, then the shortest path form is one of those:

$$(0, b-a, c-a),$$
$$(a-b, 0, c-b),$$
$$(a-c, b-c, 0),$$

and thus:

$$|D-S| = \min(|b-a| + |c-a|, |a-b| + |c-b|, |a-c| + |b-c|).$$

Notation and preliminary results (4)

- Given a packet p and its relative address (a, b, c) *in the shortest path form*,

$$\ell_p := |a| + |b| + |c|,$$

$$\ell_{max} := \max_p(\ell_p)$$

- Trivial **lower bound**:

  Any permutation routing algorithm needs at least ℓ_{max} routing steps.

Notation and preliminary results (4)

- Given a packet p and its relative address (a, b, c) *in the shortest path form*,

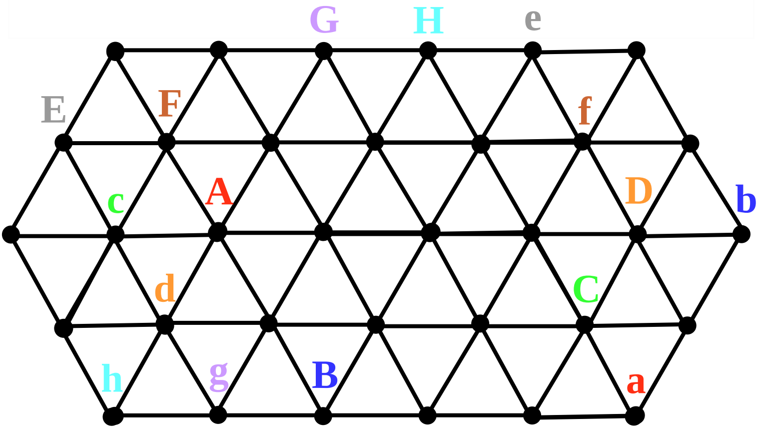
$$\ell_p := |a| + |b| + |c|,$$

$$\ell_{max} := \max_p(\ell_p)$$

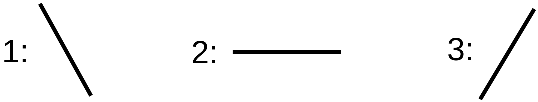
- Trivial **lower bound**:

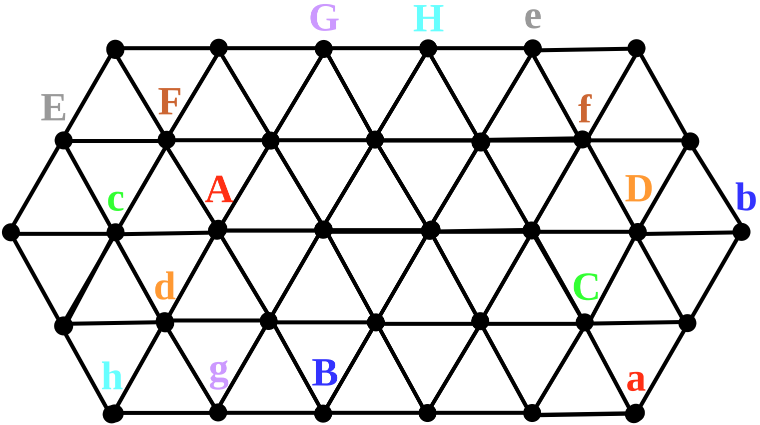
  Any permutation routing algorithm needs at least ℓ_{max} routing steps.

Example of an instance

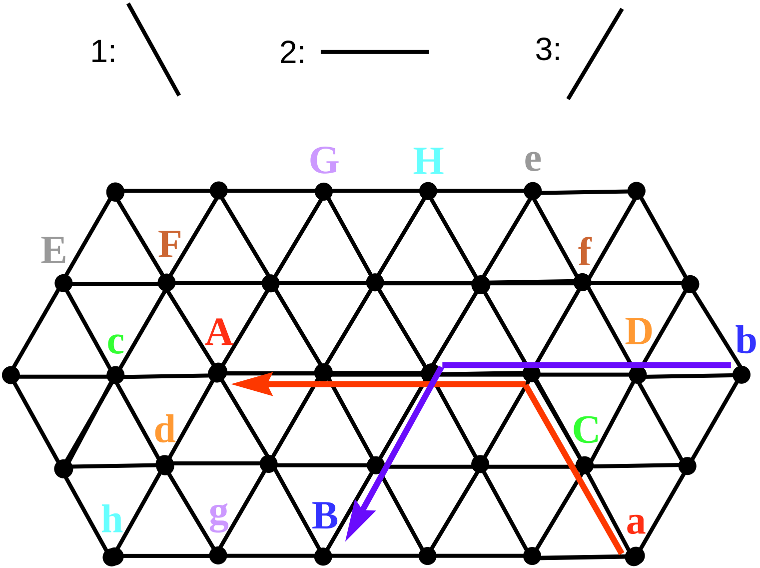


A non-optimal intuitive algorithm

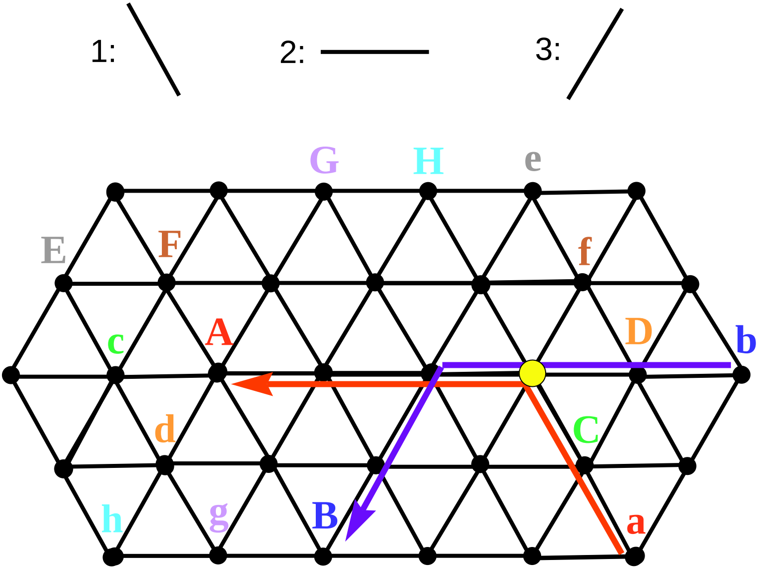




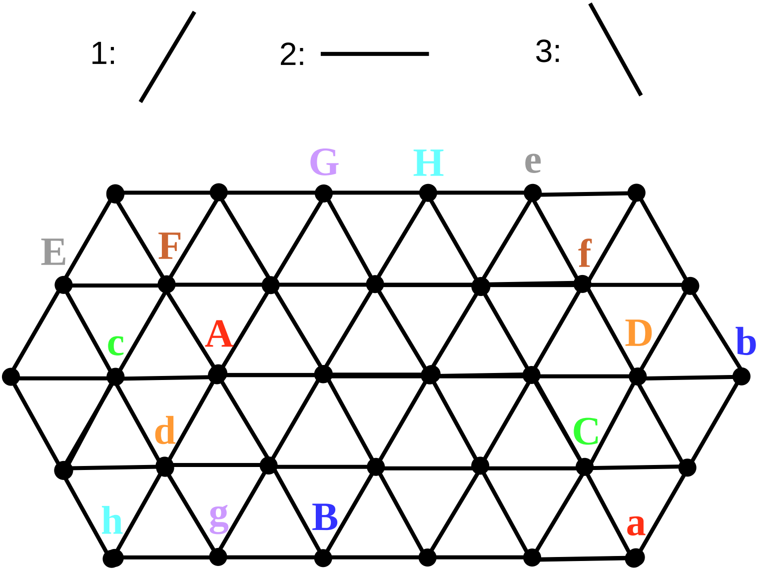
A non-optimal intuitive algorithm (2)



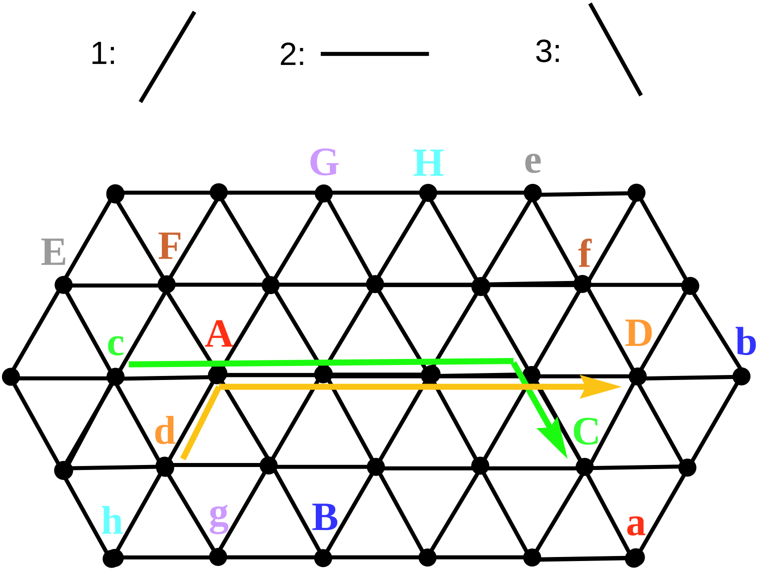
A non-optimal intuitive algorithm (3)



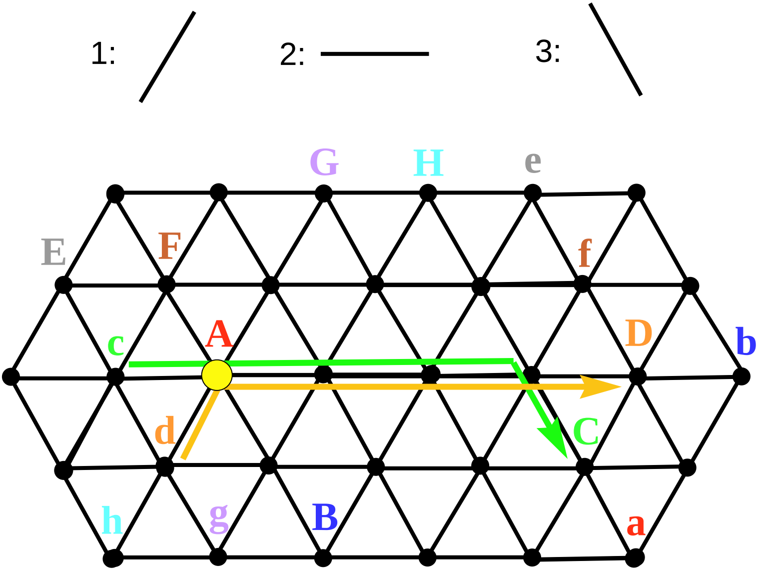
Another non-optimal intuitive algorithm



Another non-optimal intuitive algorithm (2)



Another non-optimal intuitive algorithm (3)



Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*****:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.

- ***Reception phase*****:** At each step, when a packet is received at u, its relative address is updated.

- ***Transmission phase*****:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Routing of the packets according to $\mathcal{A}$

- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)

- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$
    this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow 

  - the routing of the address $(+, -, 0)$ is represented by 

Routing of the packets according to $\mathcal{A}$

- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)
- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$ this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow

  - the routing of the address $(+, -, 0)$ is represented by

Routing of the packets according to $\mathcal{A}$

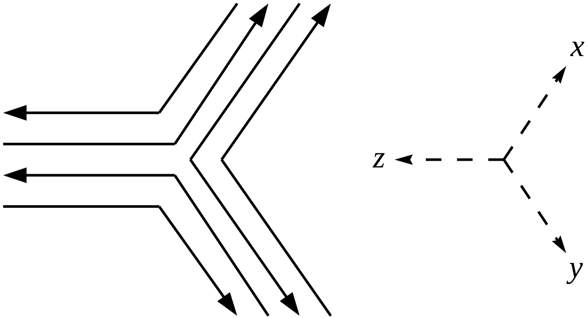
- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)
- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$
    this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow 

  - the routing of the address $(+, -, 0)$ is represented by 

Routing the packets (2)

In this figure all the routing rules are summarized:



Correctness of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

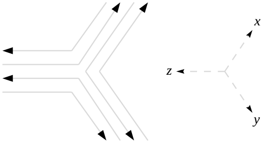
  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node → contradiction.



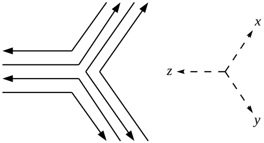
- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** → there is no ambiguity.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node $\rightarrow$ contradiction.



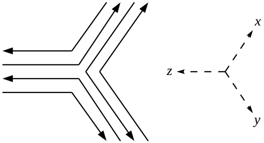
- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** $\rightarrow$ there is no ambiguity.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node $\rightarrow$ contradiction.



- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** $\rightarrow$ there is no ambiguity.

Correctness (3)

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Correctness (4)

- All the packets in in **b)** are moving along their **last direction**
  - their negative component is already finished, otherwise they would be in **a)**

- Thus, since each node is the destination of at most one packet, in **b)** *the* **packet with maximum remaining length at each outgoing edge is unique.**

Correctness (4)

- All the packets in in **b)** are moving along their **last direction**
  - their negative component is already finished, otherwise they would be in **a)**
- Thus, since each node is the destination of at most one packet, in **b)** *the* **packet with maximum remaining length at each outgoing edge is unique**.

Optimality

- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\rightarrow$ every step the maximum remaining distance over all packets decreases by one

  $\rightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.
- The only involved operation are *integer addition and comparison* among the lengths of the addresses of the packets at each node.

Optimality

- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\rightarrow$ every step the maximum remaining distance over all packets decreases by one

  $\rightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.
- The only involved operation are *integer addition and comparison* among the lengths of the addresses of the packets at each node.

Optimality

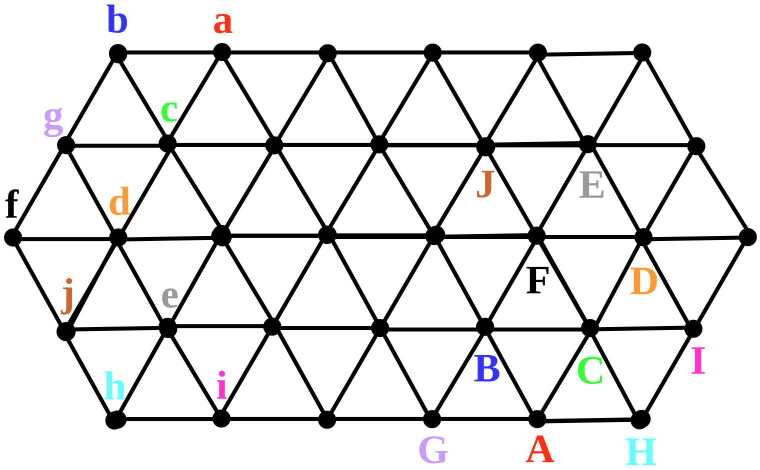
- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\rightarrow$ every step the maximum remaining distance over all packets decreases by one

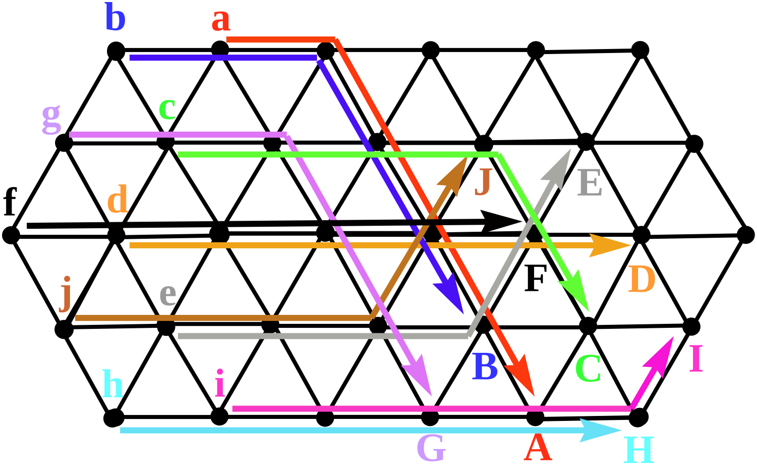
  $\rightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.
- The only involved operation are *integer addition and comparison* among the lengths of the addresses of the packets at each node.

Final example



Final example (2)



Summary

- We have solved the permutation routing problem for full-duplex triangular grids
- 2 factor approximation algorithm for half-duplex triangular grids
- 2 factor approximation algorithm for full-duplex hexagonal grids

Further research

- $(\ell - k)$-routing
- Hexagonal grid
- Half-duplex case for triangular/hexagonal grids
- Permutation routing on 3-circulant graphs
- Conceive algorithms under *average case* analysis

Thanks!