

# Direct Routing: Algorithms and Complexity

Costas Busch, RPI

Malik Magdon-Ismail, RPI

**Marios Mavronicolas, Univ. Cyprus**

Paul Spirakis, Univ. Patras

# Outline

1. **Direct Routing.**
2. **Direct Routing is Hard.**
3. **Approximation Algorithms**
  - **Arbitrary Networks.**
  - **Specific Networks.**
4. **Connection to Buffering.**
5. **Concluding Remarks.**

# Direct Routing

- Special case of bufferless routing.
- Packets are not allowed to collide.
- Once injected, packets must proceed **directly** to destination.
- Only one parameter per packet: its **injection time**.

# Direct Routing is Important

**In Practice:** When buffering is expensive or not available, for example optical networks.

**In Theory:** Gives insight into buffered routing, for example:

- Impossibility of efficient direct routing means buffering is **required**.
- How much buffering is required?

# Problem Formulation

## Given:

Packets  $\pi_i$  with pre-specified paths  $p_i$  on graph  $G$ .

- Each packet has source  $s_i$  and destination  $\delta_i$ .

## Task:

Obtain a **Direct Routing Schedule**: a time  $\tau_i$  for each packet  $\pi_i$

- if  $\pi_i$  is injected at time  $\tau_i$ , then it can proceed directly to its destination without collision.

– Since no collisions are allowed, direct routing is offline.



# Goal

1. Compute direct routing schedule in polynomial time.
2. Minimize the Direct routing time.

(1) and (2) are contradictory (in general) unless  $P = NP$ .

# Direct Routing Decision Problem

**Problem:** Direct Route

**Input:** A direct routing problem and integers  $T \geq 0$ ,

**Question:** Does there exist a direct routing schedule with maximum injection time at most  $T$ ?

**Theorem**[Direct Route is NP-Hard] There exists a polynomial time reduction from vertex coloring to Direct Route.

Further, the reduction is *gap-preserving* so Direct Route is also hard to approximate.

# Approximation Algorithms

- Near-optimal routing time  $\implies$  Near-minimal injection time.
- Near-minimal injection time is hard to obtain (NP-completeness)
- We therefore look for approximation algorithms.

# Optimal Routing Time

- A packet traverses one edge per time step.
- At most one packet can use an edge in a time step.

Let  $D$  be the maximum path length.

↑  
**Dilation**

Let  $C$  be the maximum # paths that use any edge.

↑  
**Congestion**

$$\boxed{\text{Routing Time} \geq \max\{C, D\} = \Omega(C + D)}$$

↑  
any algorithm (direct or not)

# Near-Optimal Direct Routing

We desire direct routing algorithms with near optimal, routing time,

$$O(C + D).$$

# Direct Routing Algorithms

## Arbitrary Routing Problems:

- Greedy algorithm.

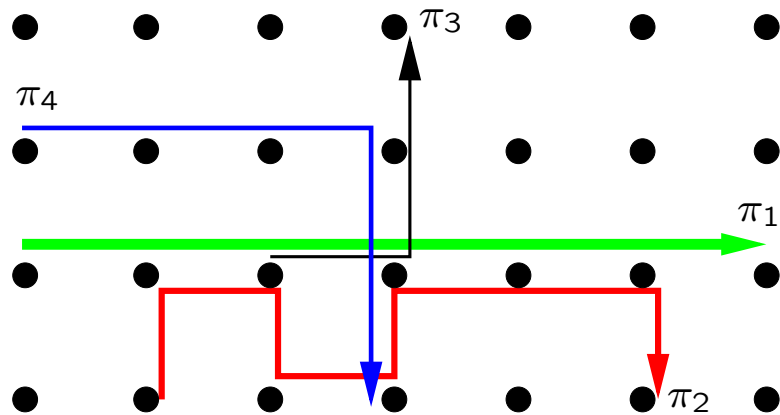
## Specific Network Topologies:

- **Trees**: Optimal direct routing with shortest paths.
- Mesh: Near Optimal with multi-bend paths.
- Butterfly: Optimal for permutations and random destinations.
- Hypercube: Optimal for permutations and random destinations

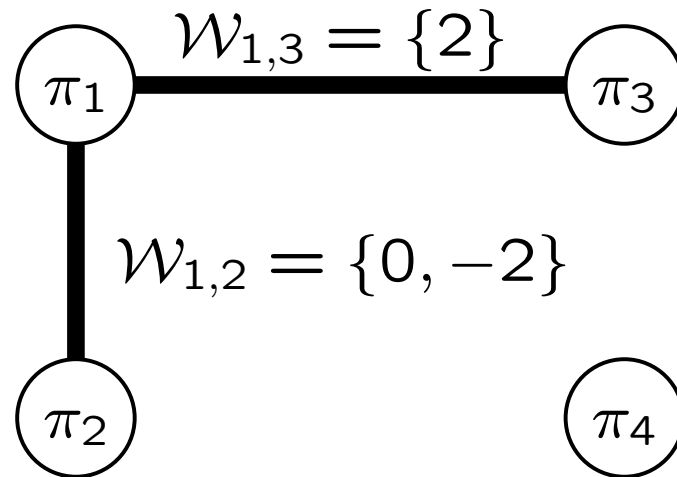
# Dependency Graphs $\mathcal{D}$

- The packets are nodes in  $\mathcal{D}$ .
- Two packets are adjacent if their paths share link in the network.
- Edges in  $\mathcal{D}$  have weights, one for each edge on which the two paths collide.
- A weight on an edge indicates that if the two packets are injected at times different by the weight, then they collide.

# Example Dependency Graph



routing problem,  $(G, \Pi, \mathcal{P})$



dependency graph,  $\mathcal{D}$

– The **weight degree** of a packet is the number of weights incident with the packet

eg.  $\pi_1$  has weight degree 3.

# Greedy Algorithm

- Arbitrarily order the packets  $\pi_1, \dots, \pi_N$
- Assign injection times in this order.
- A packet is assigned the smallest injection time available that does not cause collision with a previously assigned packet.

(Similar to greedy algorithm for vertex coloring)

# Routing Time for Greedy Algorithm

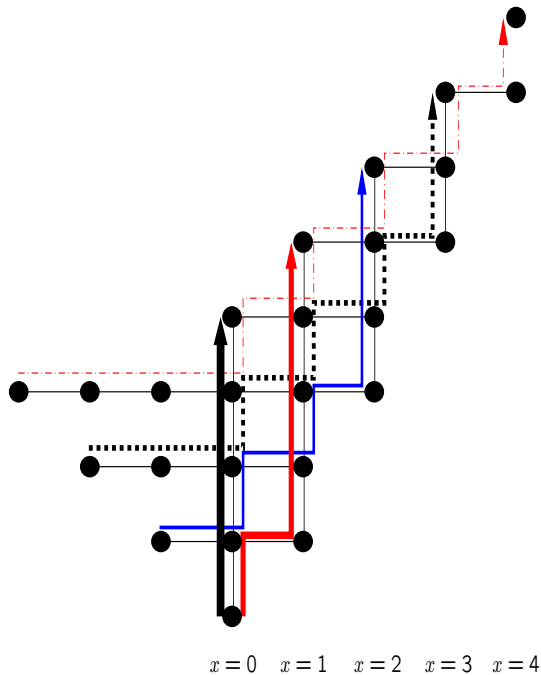
**Lemma** A packet is assigned an injection time that is at most its weight degree.

– A packet collides with at most  $C - 1$  packets per edge on its path.

**Lemma** A packets weight degree is at most  $(C - 1) \cdot D$ .

**Theorem** The routing time is at most  $C \cdot D$ .

# $C \cdot D$ Worst Case Optimal



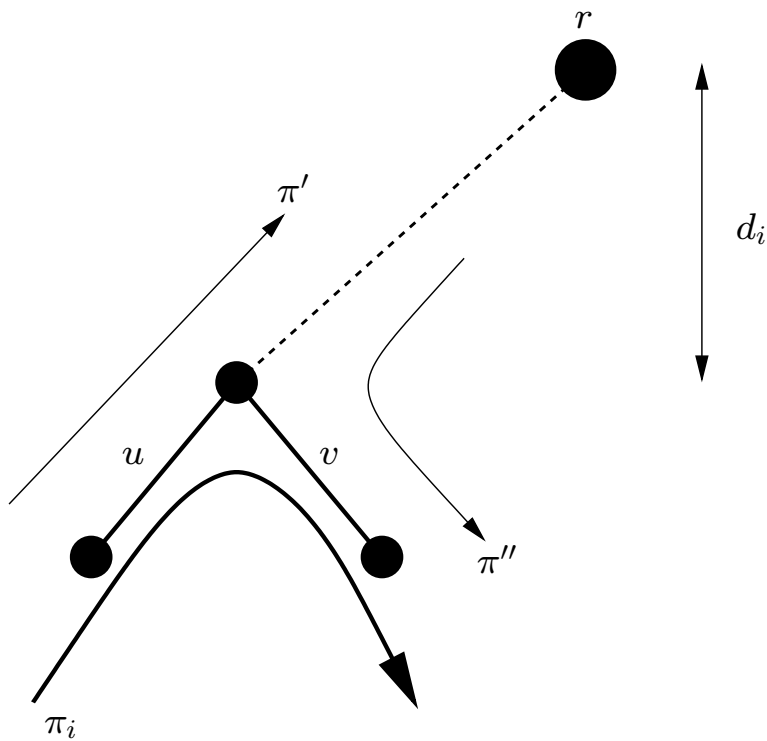
- Every two paths collide
- $\mathcal{D}$  is a clique.
- Every edge weight in  $\mathcal{D}$  is 0.
- No two packets have the same injection time.
- Thus, the latest injection time  $\geq N$ .

Send  $\sqrt{N}$  packets along each path.  $\implies C = \Theta(\sqrt{N}); D = \Theta(\sqrt{N})$ .

- Routing time is  $\Omega(C \cdot D)$ ,
- For **any** direct routing algorithm.  
(Note  $C + D = O(\sqrt{N}) = o(C \cdot D)$ .)

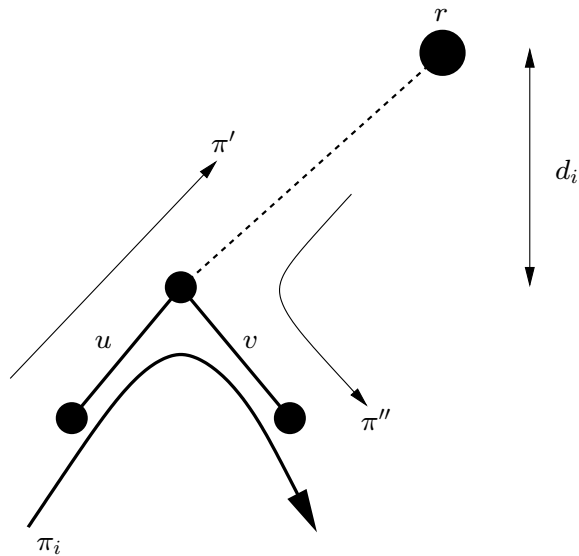
# Trees

By choosing the ordering of packets carefully, the greedy algorithm gives optimal routing time.



- Pick a root  $r$ .
- $d_i$  is the closest  $\pi_i$  gets to  $r$ .
- Sort packets according to  $d_i$ .
- Apply the greedy algorithm.

# Routing Time for Trees



- If  $\pi'$ ,  $\pi''$  occur earlier in the ordering *and* collide with then they collide either at  $u$  or  $v$ .
- This collision can be “charged” to  $u, v$ .

Thus, when  $\pi_i$  is assigned a time, at most  $2C - 2$  packets which already have a time can collide with it.

**Lemma** The injection time of  $\pi_i$  is at most  $2C - 2$ .

**Theorem** The routing time is at most  $2C + D - 2 = O(C + D)$   
**Asymptotically Optimal!**

# Mesh, Butterfly, Hypercube

## Mesh:

- Using a similar charging scheme to trees, a collision can be charged to a bend in a path.
- We show that  $\log n$  bend paths suffice ( $n$  is the network size).
- Routing time is  $O((C + D) \log n)$ . (Logarithmic factor from optimal.)

## Butterfly, Hypercube

- For random routing problems, we show that the maximum weight degree in  $\mathcal{D}$  is  $O(\log n)$ .
- Since  $D = O(\log n)$ , routing time is  $O(\log n)$ . (Worst case optimal.)

# Connection to Buffering

- There exist problems for which no direct schedule exists with near optimal routing time exists (eg.  $\Omega(C \cdot D)$  problem).
- Such problems indicate the necessity of buffering to obtain near optimal routing time.

By considering the amount of buffering required to lower the routing time for the hard problem, we get

**Theorem** There exist routing problems for which  $\Omega(N^{4/3})$  buffering is required to obtain near (within polylogarithmic factors) optimal routing time.

# Conclusions and Future Work

- For general networks, the greedy algorithm is worst case optimal.
- For specific network topologies (tree, mesh, butterfly, hypercube), direct routing is as good (routing time) as buffered routing.
- Hard direct routing problems lead to lower bounds on buffering requirements for any optimal routing algorithm.
- Online versions of direct routing?