

University of Freiburg, Germany  
Department of Computer Science

# Distributed Systems

## Chapter 5 Distributed Routing

Christian Schindelhauer

30. Mai 2013

## 5.1: Introduction

- One of the most important prevalent problems in distributed systems

### The Internet

- no central entity
- > 400,000 routers of AS (autonomous systems) use BGP (border gateway protocol) to determine routes between AS
- within AS distributed routing protocols update routing tables

### Mobile Ad-Hoc Networks

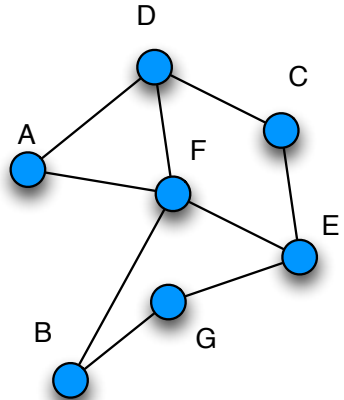
- no central control
- dynamic wireless nodes
- connections appear and disappear at any time

## 5.2: Routing in the Internet

- Packet forwarding and route selection
- Packet forwarding algorithm
  - each node has a routing table
  - when a packet needs to be processed, choose the best choice from the table
  - decrease TTL (time to live counter)
  - if  $TTL = 0$  then delete packet
- Route selection
  - programming of the routing table
  - originally: static (manually)
  - always single-path
  - flat versus hierarchical
  - intradomain and interdomain

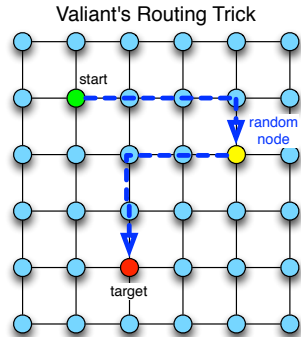
## 5.3: Routing Paradigms

- Optimization goals
  - delay
  - hop count
  - throughput
  - reliability
  - monetary cost
- Routing decisions
  - Distributed routing
    - routers decide
    - BGP, Distance-Vector, Link-State Routing
  - Source routing
    - senders decide
    - DSR (Dynamic Source Routing), Onion Routing (TOR)
  - Centralized routing
    - one (possibly external) instance decides all routing
    - (mobile) phone network, static routing



# Adaptivity

- Deterministic routing algorithm
  - optimizes the path (usually the hop count)
  - each packet is using the same route
- Oblivious routing algorithms
  - oblivious to the status of the network
  - Valiant has proved that in a two-dimensional network randomized oblivious routing optimizes the throughput
  - packets are routed in random or cyclic directions
- Adaptive routing algorithms
  - use information about network traffic / channel status
  - avoid congested areas
  - avoid unreliable links



# Switching Techniques

- Circuit Switching
  - First, determine a path and reserve the router nodes
  - Then, send the packets on this path exclusively reserved for this connection
- Packet Switching
  - store-and-forward switchings
  - each packet is *individually* routed from source to target
- Virtual cut-through switching (VCT)
  - Messages are split into packets
  - if the channel to the next router is free, the packet is immediately forwarded
  - otherwise the packets are buffered in the router with the first blockade and sent if the channel is free again
- Wormhole (WH) switching
  - like VCT but routers have small buffers
  - packet string is stored on buffers like a snake

## 5.4: Bellman Ford

- The Bellman-Ford Algorithm computes the shortest path problem towards  $t$  from each node in graph  $G = (V, E)$  with weights  $w(u, v)$

### Bellman-Ford( $G, w, s$ )

- **Init-Target**( $G, w$ )
- loop  $|V| - 1$  times:
  - for all  $(u, v) \in E$  do
    - **Relax**( $u, v$ )
- for all  $(u, v) \in E$  do
  - if  $d(u) > d(v) + w(u, v)$  then return false

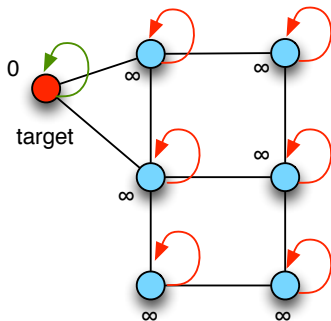
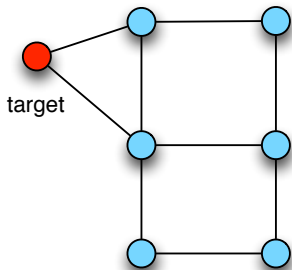
### Init-Target( $G, w, t$ )

- **Init-Target**( $G, w$ )
- for all  $v \in V$  do
  - $d(v) \leftarrow \infty$
  - $\pi(v) \leftarrow v$
- $d(t) \leftarrow 0$

### Relax( $u, v$ )

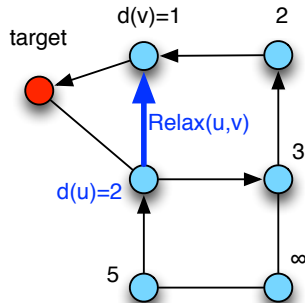
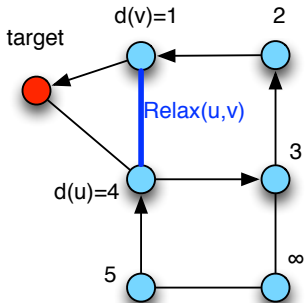
- **Relax**( $u, v$ )
- if  $d(u) > w(u, v) + d(v)$  then
  - $d(u) \leftarrow w(u, v) + d(v)$
  - $\pi(u) \leftarrow v$

# Init-Target

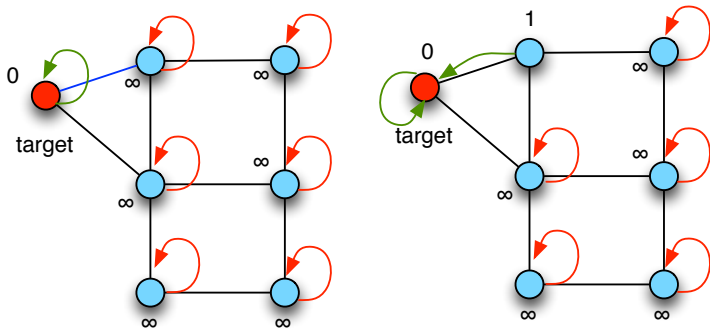




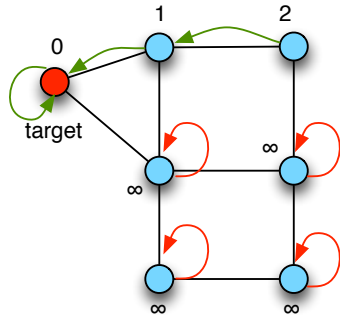
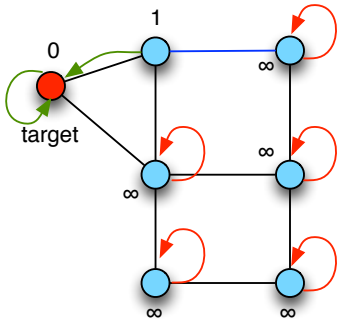
# Relax



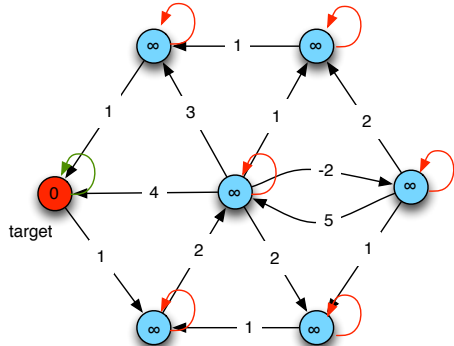
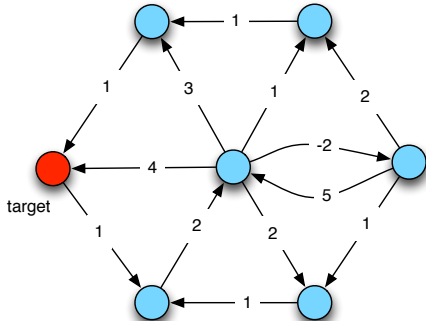
# Relax



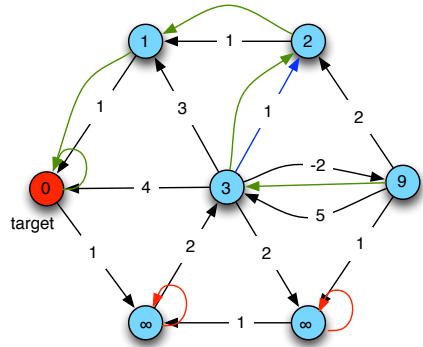
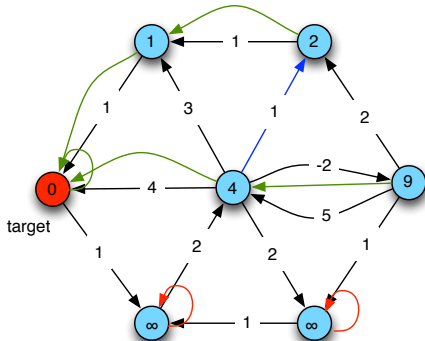
# Relax



# Init-Target for Directed Graphs



# Relax for Directed Graphs

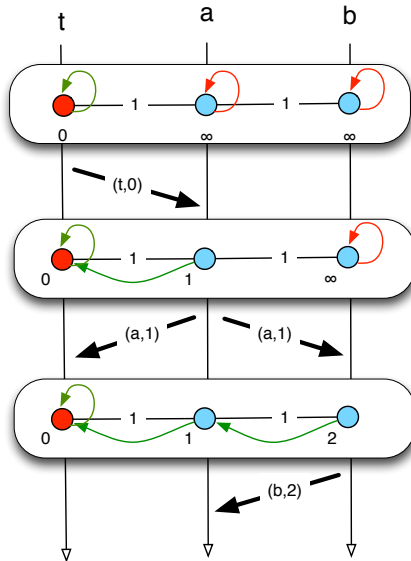


Compute for each (target) node of the network the following.

## Distributed Bellman Ford for target $t$ (Distance-Vector Routing)

- If node is  $t$  then  $d(t) \leftarrow 0$ ;  $\pi(t) \leftarrow t$
- If a message from  $u$  to  $\pi(u)$  fails then
  - $d(u) \leftarrow \infty$
- If  $u$  detects a new neighbor  $v$  then
  - send  $(u, d(u))$  to  $v$
- If  $u$  receives  $(v, d(v))$  from  $v$ 
  - if  $d(u) > d(v) + w(u, v)$  or  $v = \pi(u)$  then
    - $d(u) \leftarrow d(v) + w(u, v)$
    - $\pi(u) \leftarrow v$
- if  $d(u) = \infty$  then  $\pi(u) \leftarrow u$
- Every time  $d(u)$  or  $\pi(u)$  has changed  $u$  sends  $(u, d(u))$  to all neighbors

# Distance Vector



# Correctness

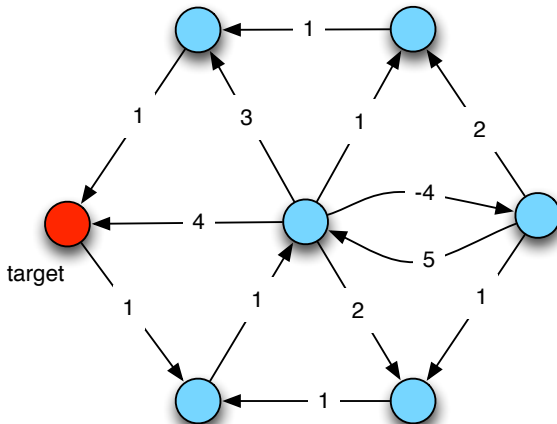
- Let  $|u, v|$  be the distance from  $u$  to  $v$
- Assume that the weights are constant
- Then, at each time of the operation  $d(u) > |u, t|$
- If the shortest path from  $u$  to  $t$  is  $(u, v_1, v_2, \dots, v_n, t)$  and
  - a message is sent from  $t$  to  $v_n$  and then from  $v_n$  to  $v_{n-1}$ , etc.
  - then  $d(v_n) = |v_n, t|$ ,  $d(v_{n-1}) = |v_{n-1}, t|$ ,  $\dots$ ,  $d(u, t) = |u, t|$
- So, for each shortest path of finite length eventually Distributed Bellman Ford converges



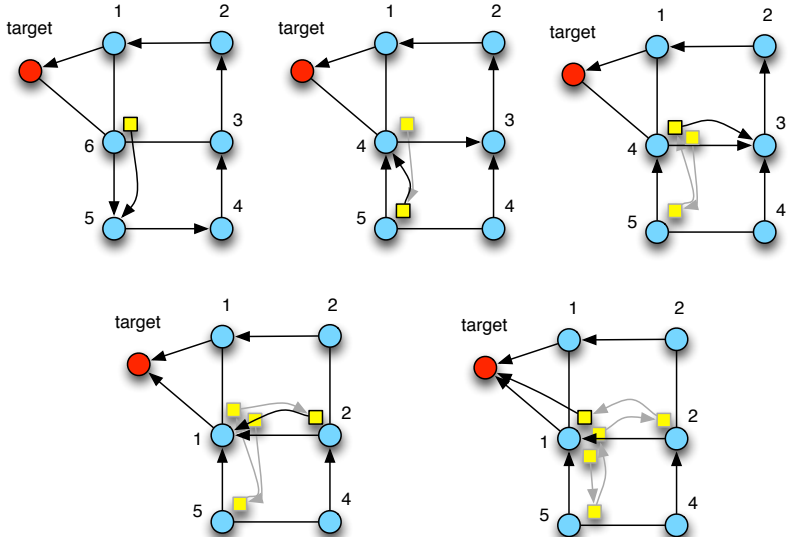
# Problems of Distributed Bellman Ford

- Negative path cycles
  - Bellman-Ford works fine as long as the shortest path is finite
  - if a negative path cycle exist, then the shortest path is infinite
- Dynamic graphs
  - Temporarily wrong routes
    - During the distributed computation messages might take wrong directions
    - could even revisit the same node more than once
  - Lost connections
    - Bad news travels slowly (or not at all)
    - If the distance increases then no messages are produced
    - Old an wrong information is preserved
- Temporarily undefined routes
  - For speeding up store last message of each neighbor
  - Use this information when new messages arrive
- Count-to-Infinity Problem
  - If the target is not reachable and at least two nodes remain connected
  - then the distance is updated towards infinity

# Negative Path Cycles

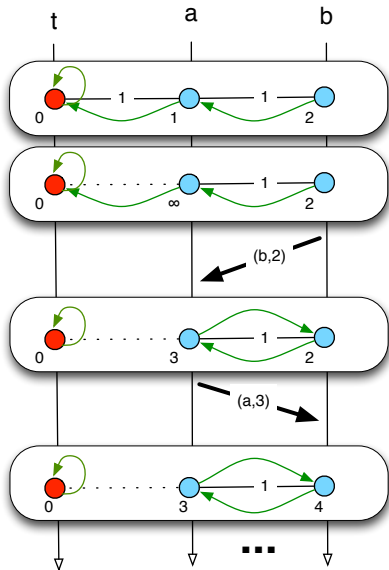


# Temporarily Wrong Routes



# Count to Infinity Problem

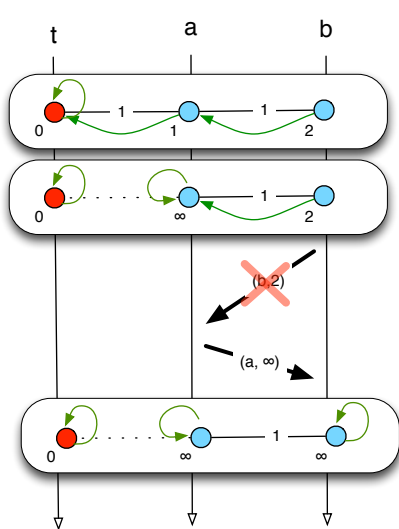
- If the target is not reachable and at least two nodes remain connected
- then the distance is updated towards infinity



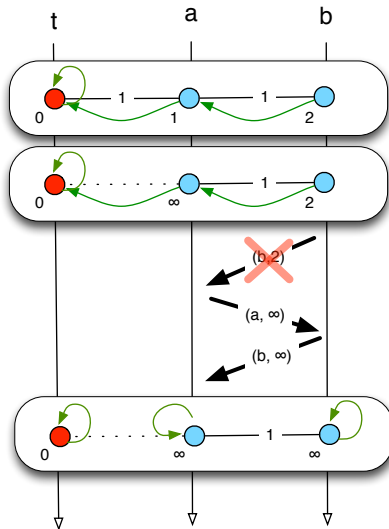
# Solutions to the Count-to-Infinity Problem

- Split Horizon
  - A node does not advertise routes to nodes on its path to the target
- Poison reverse
  - A variant of split horizon
  - A node does not advertise routes to nodes on its path to the target
  - With the exception of a value of  $\infty$  is advertised immediately towards target
- Link State Routing
  - all link information is broadcast in the network
  - each node computes the shortest path on this information
- Path Vector Protocol
  - instead of only storing the next link
  - store the complete path and forward it to the neighbors
  - used in BGP

## Split Horizon



## Poison Reverse



## Distance-Vector Routing with Split Horizon for target $t$

- If node is  $t$  then  $d(t) \leftarrow 0$ ;  $\pi(t) \leftarrow t$
- If a message from  $u$  to  $\pi(u)$  fails then
  - $d(u) \leftarrow \infty$
  - $\pi(u) \leftarrow u$
- If  $u$  detects a new neighbor  $v$  then
  - send  $(u, d(u))$  to  $v$
- If  $u$  receives  $(v, d(v))$  from  $v$ 
  - if  $d(u) > d(v) + w(u, v)$  or  $v = \pi(u)$  then
    - $d(u) \leftarrow d(v) + w(u, v)$
    - $\pi(u) \leftarrow v$
- Periodically and every time  $d(u)$  or  $\pi(u)$  has changed  $u$  sends  $(u, d(u))$  to all neighbors **except for  $\pi(u)$**

Split horizon rule: information is not sent towards target

## Distance-Vector Routing with Poison Reverse for target $t$

- If node is  $t$  then  $d(t) \leftarrow 0$ ;  $\pi(t) \leftarrow t$
- If a message from  $u$  to  $\pi(u)$  fails then
  - $d(u) \leftarrow \infty$
  - $\pi(u) \leftarrow u$
- If  $u$  detects a new neighbor  $v$  then
  - send  $(u, d(u))$  to  $v$
- If  $u$  receives  $(v, d(v))$  from  $v$ 
  - if  $d(u) > d(v) + w(u, v)$  or  $v = \pi(u)$  then
    - $d(u) \leftarrow d(v) + w(u, v)$
    - $\pi(u) \leftarrow v$
- Periodically and every time  $d(u)$  or  $\pi(u)$  has changed send  $(u, d(u))$  to all neighbors **except for  $\pi(u)$**
- **If  $d(u)$  has changed to  $\infty$  then send  $(u, d(u))$  to  $\pi(u)$**   
**Poison reverse: remove loops before they can propagate**

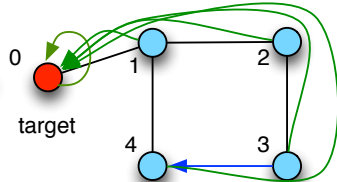
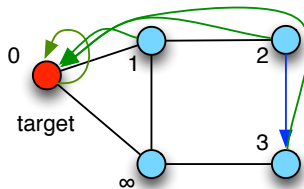
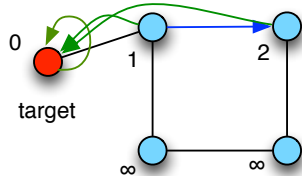
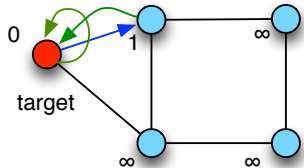


## Path Vector Protocol for target $t$

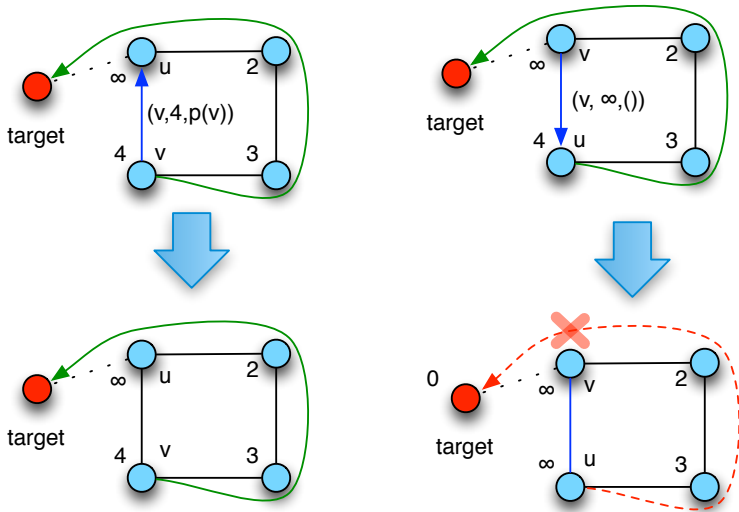
- $d(t) \leftarrow 0$ ;  $p(t) \leftarrow (t)$
- $t$  send  $s(t, d(t), p(t))$  to all neighbors once
- If a message from  $u$  to the first node of  $p(u)$  fails then
  - $d(u) \leftarrow \infty$
  - $p(u) \leftarrow ()$
- If  $u$  detects a new neighbor  $v$  then
  - send  $(u, d(u), p(u))$  to  $v$
- If  $u$  receives  $(v, d(v), p(v))$  from  $v$ 
  - if  $d(v) = \infty$  and  $v \in p(u)$  then  $u$ 's route has vanished
    - $d(u) \leftarrow \infty$
    - $p(u) \leftarrow ()$
  - if  $u \notin p(v)$  and  $d(u) > d(v) + w(u, v)$  then only if  $u$  is not in the path to  $t$  relax distance
    - $d(u) \leftarrow d(v) + w(u, v)$
    - $p(u) \leftarrow (u, p(v))$  append  $u$  to path from  $v$  to  $t$
- Periodically and every time  $d(u)$  or  $p(u)$  has changed send  $(u, d(u), p(u))$  to all neighbors



# Relax in Path Distance Vector



# Lost Target in Path Distance Vector



# Applications of Routing Algorithms

- Mobile Ad Hoc Networks
  - reactive versus proactive protocols
  - proactive protocols like DV construct routing tables even without packets
  - when the network is too dynamic this is an overhead
  - then reactive protocols like Flooding, DSR, AODV are more adequate
- DV routing is used in Intra-Domain Routing
  - RIP (Routing Information Protocol)
  - EIGRP (Enhanced Interior Gateway Routing Protocol)
- main alternative for Intra-Domain Routing is Link-State-Routing used in
  - OSPF (Open Shortest Path First)
  - IS-IS (Intermediate System to Intermediate System Protocol)
- Path-Vector-Protocol is the worldwide standard in Inter-Domain Routing
  - BGP (Border Gateway Protocol)

End of Section 5