NSWI090: **Computer Networks**

http://www.ksi.mff.cuni.cz/~svoboda/courses/242-NSWI090/

Lecture 5

Routing

Martin Svoboda
martin.svoboda@matfyz.cuni.cz

19. 3. 2025

Charles University, Faculty of Mathematics and Physics

Lecture Outline

Network layer

- **Routing** and **forwarding** tasks
  - Basic concepts
  - Classification of routing approaches

Routing approaches

- Static Routing, Flooding, …
- Backward Learning, Source Routing, …
- Distance-Vector Routing
- Link-State Routing
- Path-Vector Routing
- …

Network Layer

Network layer tasks

- Delivery of packets across a **system of networks** mutually interconnected by **routers** to the intended **recipient** node

Routing

- Process of **finding optimal delivery routes**
  - Maintenance of **routing tables**
    - Capture **network topology** and other necessary information
  - Calculation of the actual routing paths
    - **Searching for the shortest paths in weighted multi-graphs**

Forwarding

- Process of the actual **delivery of packets**
  - Based on **forwarding tables**

Routing and Forwarding Tables

Routing table

- **List of routing records** with the following **fields**
  - **Destination**: identifier of the **target network**
    - E.g.: 192.168.2.0 with netmask 255.255.255.0
  - **Interface**: local **interface to be used**
    - E.g.: 192.168.1.123
  - **Gateway**: **neighboring router** forming the first hop
    - E.g.: 192.168.1.1
  - **Metric**: **cost estimate** of reaching the target network
    - E.g.: 11

Forwarding table

- Compact structure with **already resolved routes**
  - Allows for **efficient forwarding**

Common Principles

Destination-based routing

- Routing is only based on the **recipient address**
  - I.e., source address is not considered

Least-cost routing

- Optimal route is chosen according to the **lowest cost**

Hop-by-hop routing

- Routers make their decisions locally and on their own
  - I.e., they are **independent on other routers** on the way

Content-independent routing

- Contents and character of data are not taken into account

Stateless routing

- Decisions are **independent on history** and previous datagrams

Routing Classification

Basic **classification** of routing approaches

- Non-adaptive / adaptive
  - Whether **network changes** are detected and reflected
- Centralized / distributed
  - Whether **routing decisions** are made by independent routers
- Isolated / non-isolated
  - Whether **mutual cooperation** of routers is expected
- Interior / exterior
  - What is the **deployment scope** within hierarchical routing

Routing Classification

Other aspects to consider

- Classful / classless
  - Whether only **legacy IP addresses with classes** are assumed
    - Or netmasks or CIDR prefixes are supported
- Unicast / anycast / multicast / broadcast
  - What **kind of transmissions** is assumed
- ...

Routing Strategies

Adaptive **routing** (dynamic routing)

- **Capable of adapting to network changes**
  - Such as changes in **network topology**, **traffic load**, …
  - **Routing tables** are constructed and **updated dynamically**
    - And so routing decisions may change in time
- More complex, used more often in practice
  - Causes considerable challenges especially in large systems

Non-adaptive **routing** (static routing)

- **Does not adapt to changes** nor cooperate with other nodes
  - **Routing tables** (if any) **are fixed** and given in advance
- Suitable in specific situations only
- Examples: Fixed Directory Routing, Random Walk, Flooding

Non-Adaptive Routing

Fixed Directory Routing (Static Routing)

- **Routing tables are configured manually by administrators**
  - Routing records do not change in time
- Advantages
  - **Exact paths** are given and **known in advance**
  - Higher level of achieved **security**
    - Since no update information is disseminated, it cannot be faked
  - **Specific requirements** can easily be handled
- Disadvantages
  - Insensitive to changes $\Rightarrow$ **cannot recover from failures**
  - Too **tedious** in large and complex networks
  - Administrators can make unintentional **mistakes**

Non-Adaptive Routing

Fixed Directory Routing (cont'd)

- **Combinable with adaptive approaches**
  - **Default route**
    - Exit direction when no other routes are available or necessary
  - **Failsafe backup**
    - In case dynamic routing becomes unavailable,
      static routes can take precedence

Random Walk Routing (Random Routing)

- **Incoming packet is sent to a <u>randomly</u> chosen neighbor**
  - Different to the one it arrived from
- Use cases
  - Only when the probability of reaching the destination is high
    - Peer-to-peer (P2P) networks

Flooding

Flooding (Flood Routing)

- Incoming **packet is duplicated and sent to all directions**
  - All except the one it arrived from
  - I.e., **no routing tables** are used
- Advantages and disadvantages
  - Requires **no network information**
  - Very **simple to implement**
  - **Always successful** if path exists
  - Duplication **increases the load**, though
- Use cases
  - Whenever **high robustness** is required
    - E.g., emergency messages, military applications, …
  - L2 **local broadcast**

Flooding

Issue: topologies with **loops**

- **One loop** exists…
  - Already sent packets can once again return
- Two or even **more loops** exist…
  - Packets will get duplicated repeatedly (broadcast storm)

⇒ **recurring packets** need to be <u>identified</u> and then <u>eliminated</u>

- **Uncontrolled flooding**
  - Does not prevent from indefinite recirculation at all
    - I.e., no precautions are taken
- **Controlled flooding** (selective flooding)
  - Techniques allowing to overcome the impact of loops
    - Hop Count, Sequence Numbers, Spanning Tree, …
    - They can all be used together with adaptive routing, too

Controlled Flooding Techniques

Hop Count

- Each packet contains a **counter**
  - Its **initial value** is set by the sender
  - Must be high enough
    - Otherwise the intended recipient may not be reachable
  - Network **diameter** can be used
    - When no better estimate is available
- Counter is **decremented** at each hop
- Packet is **discarded** when the counter becomes zero

Controlled Flooding Techniques

Sequence Numbers

- Each packet contains a **sequence number**
  - Assigned sequentially by the sender
- **List of sender address / sequence number pairs** is kept
  - Repeatedly encountered packets are ignored
- Issues
  - **Available space** is always limited, it can be depleted
  - Sender can shutdown and reconnect, sequence gets restarted
  - ⇒ **new packets can wrongly be recognized as old ones**
- Alternatives
  - **Packet itself** or its **checksum** can be remembered
- Example: Sequence Number Controlled Flooding (SNCF)

Controlled Flooding Techniques

Reverse Path Forwarding

- **Packet is forwarded only if it comes from the same direction that would normally be used** to reply to a given sender
  - If this direction is not provided by dynamic routing, it can be remembered the first time we come across a given sender
- Example: Reverse Path Forwarding (RPF)

Spanning Tree

- **Spanning tree** is created first
  - Minimal connected subgraph with all the nodes (and so without loops)
- **Packet is only forwarded along the links forming the tree**
- Example: Spanning Tree Protocol (STP)

Adaptive Routing

Adaptive routing

- **Routing tables** and decisions are **adaptively updated**
  - Based on **network topology**, **path costs** or **traffic load** changes
- **Ultimate goal**: **routing convergence**
  - Process leading to the **state of fully operating system** when **all routers have the same perception** of the reality
    - Information they gathered must not be mutually inconsistent
    - Must reflect the real state of the network

Possible strategies

- **Distributed routing**
  - Each router makes routing decisions independently on its own
- **Centralized routing**
  - Routing decisions are solely made by one centralized authority

Centralized Routing

Centralized **routing**

- **Routing decisions** are made by one **centralized authority**
  - So called route server
- **Other nodes** perform **forwarding** only
  - Names vary: edge device, multilayer switch
  - Whenever routing information is not yet known, **routing request** is sent to the route server
    - Its decision is **remembered**…
    - … and intentionally **forgotten** after a certain period of time
- Advantages and disadvantages
  - Route server has **full knowledge**
    - And so routing can be complex and **flexible**
  - However, it represents a **single point of failure**
    - Its failure impacts everything

Distributed Routing

Distributed routing

- Each router is eligible for making routing decisions on its own

Possible strategies

- **Isolated routing**
  - **Nodes do not cooperate** with each other at all
    - Routing solely depends on the information they locally have
  - Examples: Backward Learning, Source Routing, Hot Potato
- **Non-isolated routing**
  - **Nodes do mutually cooperate**
    - They at least interchange **available routing information**
    - They can also interact on **distributed routing calculations**
  - Examples: Distance-Vector, Link-State, Path-Vector
    - Represent core Internet routing strategies

Isolated Routing

Backward Learning

- Routing table is empty at the beginning
- Whenever a **packet from an unknown sender** is received
  - Direction of this sender is remembered
- **Incoming packet is forwarded**…
  - To **all directions** in case a given **recipient is not yet known**
    - As if **flooding** mechanism is exploited
  - Just to the single **remembered direction** otherwise
- Requirements
  - Stored information must be **periodically forgotten**
    - So that we can adapt to changes in the network
  - **Loops** must be treated appropriately

Isolated Routing

Backward Learning (cont'd)

- Possible improvement
  - **Hop counters** can be incorporated
    - Each packet contains a counter that is **incremented** at each hop
    - When a new **path with lower cost** is discovered, the currently remembered direction is **updated**
- Disadvantages
  - **Unacceptably slow convergence** in larger systems
    - Cannot be used for routing at L3 at all
- **Real-world deployment at L2**
  - **Ethernet**
    - **Forwarding of frames within complex local networks**
    - Learning process is fast enough (since the scope is limited)
    - Allow bridges / switches to be used as Plug&Play devices

Isolated Routing

Source Routing (Path Addressing)

- Basic principle
  - **Sender is responsible for finding the complete routing path**
    - Modeled as a **sequence of addresses of individual routers**
  - Once found, it is then used for the actual data
- **Discovery phase**
  - Special **explorative packet** is first sent using **flooding**
    - Each router appends the gradually built sequence by its address
  - Sooner or later one packet copy reaches the intended recipient
  - It then sends the **fully recognized path** back to the sender
- **Transmission phase**
  - Each packet is **equipped with the resolved intended sequence**
    - Individual routers simply follow this sequence when forwarding

Isolated Routing

Source **Routing** (cont'd)

- Alternatives
  - Routing path may be determined **completely** or **partially** only
- Advantages
  - **Always finds the shortest path** (if any)
    - Alternative paths can actually be found as well
    - But only one particular can be prescribed
- Disadvantages
  - **Flooding is needed** with all its cons
  - All **routers** on the way **must cooperate**
- **Real-world deployment** once again at L2
  - **Token Ring**
    - Based on a ring logical topology over a star physical topology

Isolated Routing

Hot Potato **Routing**

- We have no routing table, nor we are attempting to create it
- **Incoming packet is forwarded to the least busy direction**
  - I.e., its **output queue is the shortest one**
    - Relatively to the transmission capacity of a given path
- Disadvantage
  - Chance that this direction will be the right one is, of course, low
- **Real-world usage**
  - Temporary strategy in the event of **approaching capacity limits**
    - So that we **try to avoid router congestion**
      by getting rid of packets as fast as we can
  - Similarly at L2

Non-Isolated Routing

Non-isolated distributed adaptive routing

- In a nutshell…
  - **Adaptive** = capable of responding to network changes
  - **Distributed** = decisions are made by independent routers
  - **Non-isolated** = these routers cooperate with each other
    - The question is to what extent…
    - Differences between the existing approaches are significant
- Essential requirement
  - **Interchange of necessary routing information**
    - So that routers can inform each other about network changes
    - And so that their own **routing tables** can be updated
  - $\Rightarrow$ suitable **protocols** are needed
    - RIP, OSPF, BGP, …

Non-Isolated Routing

Distance-Vector **Routing**

- Each node only has a **partial information** on network topology
  - And so **distributed calculation** of routing paths is involved
    - I.e., so far discovered routes are **incrementally refined**
- Example: **RIP** (Routing Information Protocol)

Link-State **Routing**

- Each node has a **full knowledge** of network topology
  - And so each node can make **individual calculations** on its own
- Example: **OSPF** (Open Shortest Path First)

Path-Vector **Routing**

- Later on…

Distance-Vector Routing

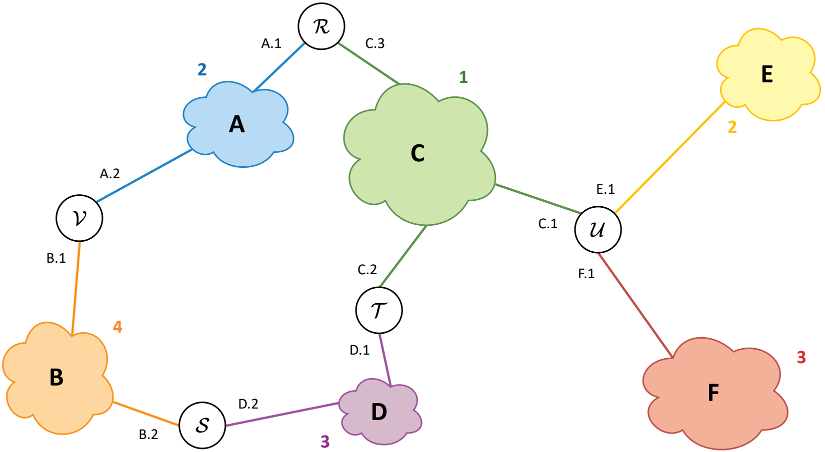
Distance-Vector Routing

- Each node maintains its own **routing table**
  - With the shortest resolved route to each discovered network
- These tables are **mutually interchanged**
  - Sooner or later **convergence** is attained

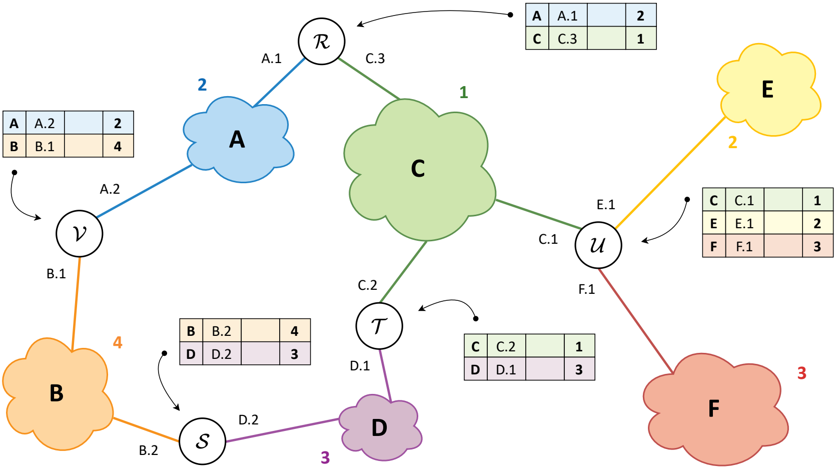
Distance vector = **routing table** with the following fields

- Destination: identifier of the **target network** to be reached
- Direction: local **interface to be used** for this purpose
- Gateway: neighboring **router to be contacted**
  - Omitted in case of direct forwarding within our network
- Metric: **overall cost** of reaching the target network
  - Generic cost, hop count, bandwidth, delay, …

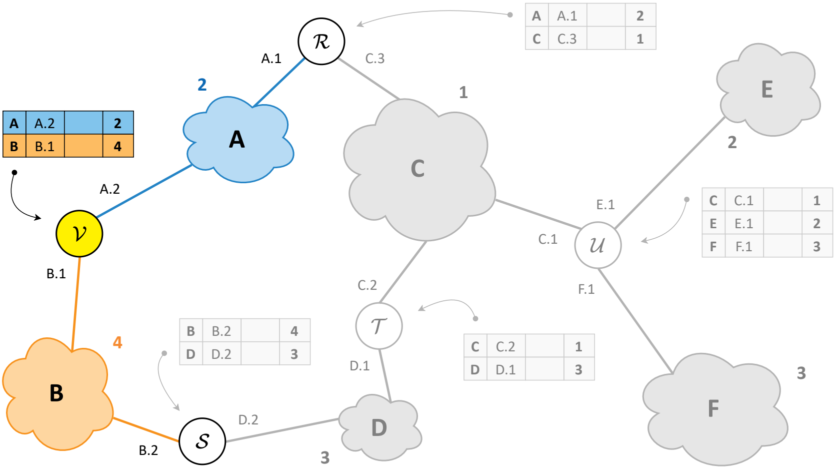
Distance-Vector Routing



Distance-Vector Routing



Distance-Vector Routing

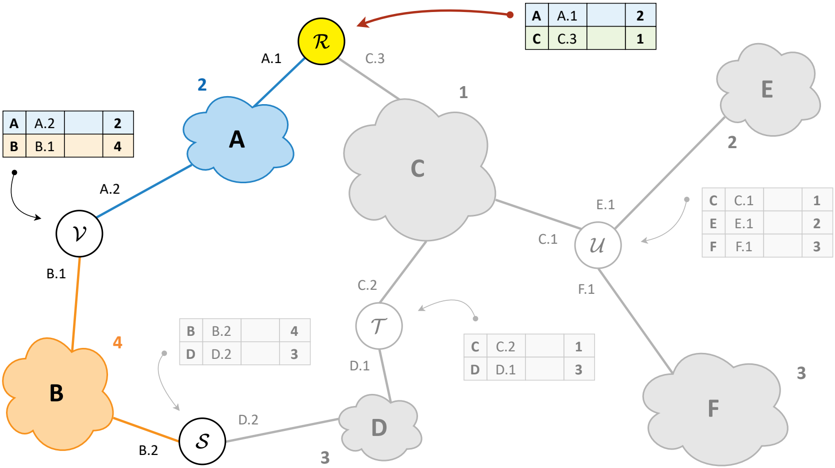


Distance-Vector Routing

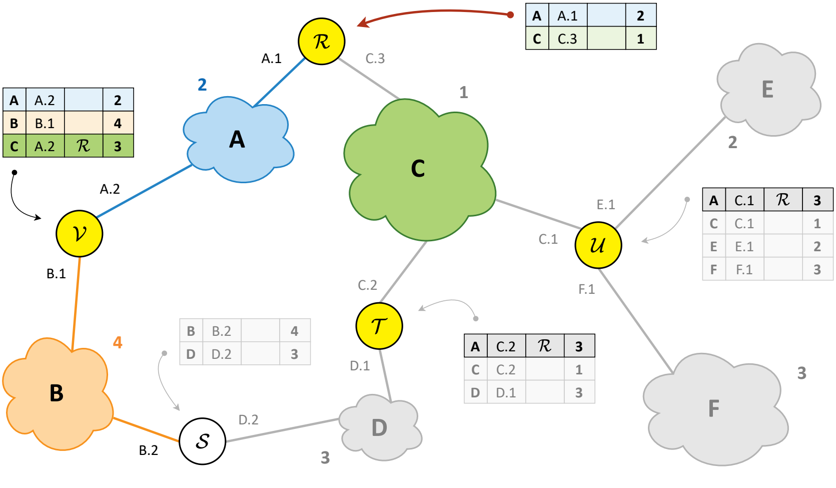
Convergence process

- **Initial routing table** is constructed
  - Only **directly reachable local networks** are included
    - Via L2 interfaces a given node has
- Tables are then **regularly interchanged between neighbors**
  - Only immediate neighbors are involved!
  - Entire process is **asynchronous**
    - I.e., individual nodes are not mutually synchronized, they act independently on each other
  - **Time interval is relatively short**
    - E.g., just 30 seconds
- **Whenever an advertised table is received** from our neighbor
  - It is used for the **refinement of our own routing table**

Distance-Vector Routing



Distance-Vector Routing



A	A.1		2
C	C.3		1

A	A.2		2
B	B.1		4
C	A.2	$\mathcal{R}$	3

B	B.2		4
D	D.2		3

A	C.2	$\mathcal{R}$	3
C	C.2		1
D	D.1		3

A	C.1	$\mathcal{R}$	3
C	C.1		1
E	E.1		2
F	F.1		3

Distance-Vector Routing

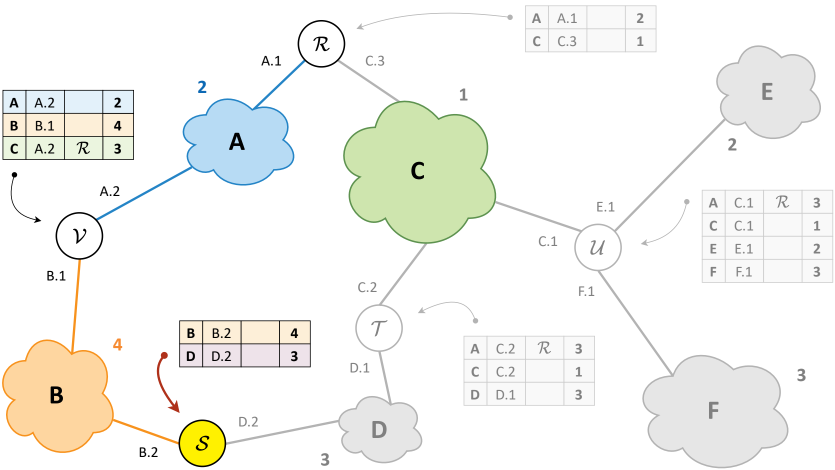
Refinement principle

- Let us assume that node X can directly reach a neighboring node Y with cost $c_{X \to Y}$
- Whenever Y advertises that it can reach network N with an overall cost $c_{Y \to N}$, we can conclude that X can also reach N, in particular via Y, with overall cost $c_{X \to N} = c_{X \to Y} + c_{Y \to N}$
  - The question is, whether this observation should be exploited
  - I.e., whether it leads to something new or better

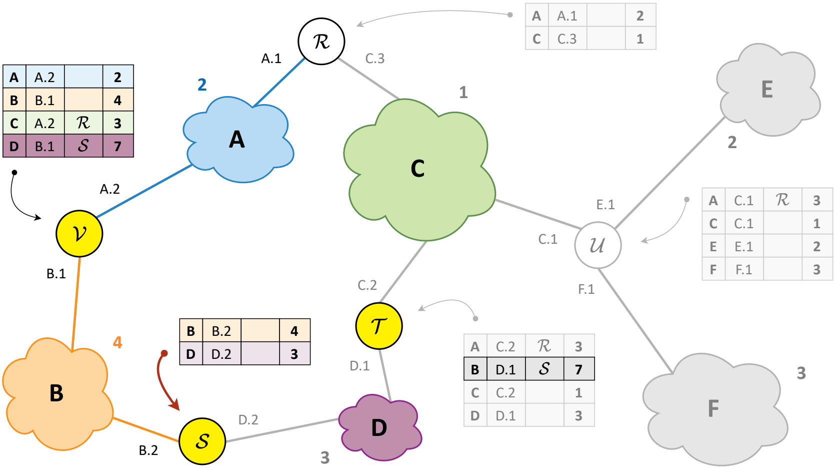
Formal background

- Distributed variation of **Bellman-Ford algorithm**
  - Allows to find shortest paths from a single source vertex to all other vertices in a given weighted graph

Distance-Vector Routing



Distance-Vector Routing



A	A.1		2
C	C.3		1

A	A.2		2
B	B.1		4
C	A.2	$\mathcal{R}$	3
D	B.1	$\mathcal{S}$	7

A	C.1	$\mathcal{R}$	3
C	C.1		1
E	E.1		2
F	F.1		3

B	B.2		4
D	D.2		3

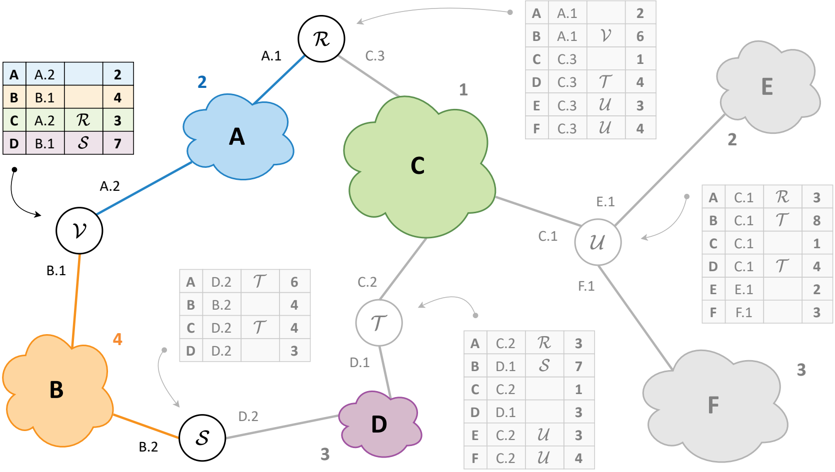
A	C.2	$\mathcal{R}$	3
B	D.1	$\mathcal{S}$	7
C	C.2		1
D	D.1		3

Distance-Vector Routing

Complete **refinement rules**

- If N is **so far unknown**
  - New record for N via Y is created
- Else if N is **already reachable via the same node** Y
  - The current record is preserved an only its cost is updated
    - Even in case the new cost is worse
- Else if N is **already reachable but via a different neighbor** and the **new cost is better**
  - The current record is fully replaced with the new option via Y
- **Otherwise nothing is updated**

Distance-Vector Routing



A	A.2		2
B	B.1		4
C	A.2	$\mathcal{R}$	3
D	B.1	$\mathcal{S}$	7

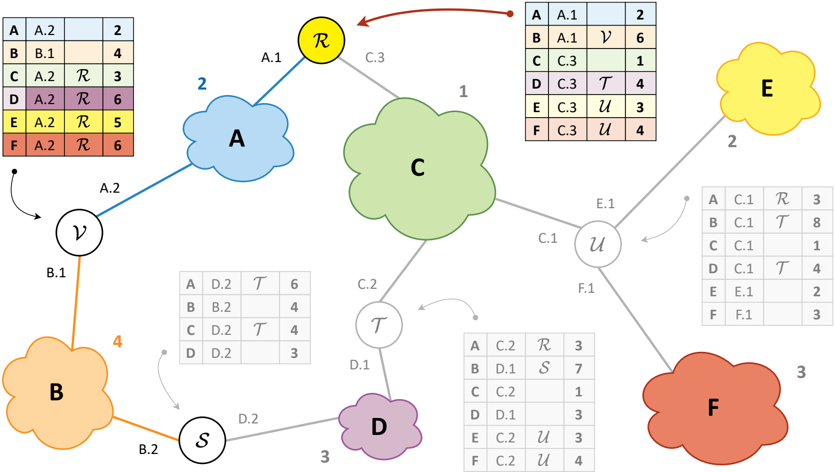
A	A.1		2
B	A.1	$\mathcal{V}$	6
C	C.3		1
D	C.3	$\mathcal{T}$	4
E	C.3	$\mathcal{U}$	3
F	C.3	$\mathcal{U}$	4

A	C.1	$\mathcal{R}$	3
B	C.1	$\mathcal{T}$	8
C	C.1		1
D	C.1	$\mathcal{T}$	4
E	E.1		2
F	F.1		3

A	D.2	$\mathcal{T}$	6
B	B.2		4
C	D.2	$\mathcal{T}$	4
D	D.2		3

A	C.2	$\mathcal{R}$	3
B	D.1	$\mathcal{S}$	7
C	C.2		1
D	D.1		3
E	C.2	$\mathcal{U}$	3
F	C.2	$\mathcal{U}$	4

Distance-Vector Routing



A	A.2		2
B	B.1		4
C	A.2	$\mathcal{R}$	3
D	A.2	$\mathcal{R}$	6
E	A.2	$\mathcal{R}$	5
F	A.2	$\mathcal{R}$	6

A	A.1		2
B	A.1	$\mathcal{V}$	6
C	C.3		1
D	C.3	$\mathcal{T}$	4
E	C.3	$\mathcal{U}$	3
F	C.3	$\mathcal{U}$	4

A	C.1	$\mathcal{R}$	3
B	C.1	$\mathcal{T}$	8
C	C.1		1
D	C.1	$\mathcal{T}$	4
E	E.1		2
F	F.1		3

A	D.2	$\mathcal{T}$	6
B	B.2		4
C	D.2	$\mathcal{T}$	4
D	D.2		3

A	C.2	$\mathcal{R}$	3
B	D.1	$\mathcal{S}$	7
C	C.2		1
D	D.1		3
E	C.2	$\mathcal{U}$	3
F	C.2	$\mathcal{U}$	4

Distance-Vector Routing

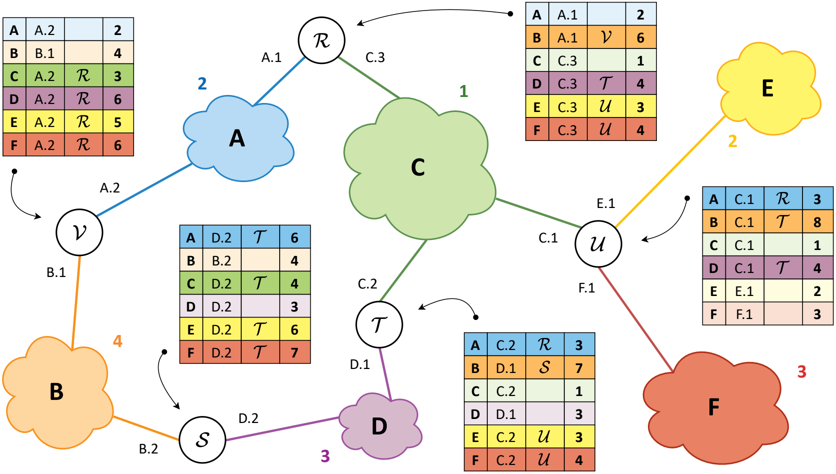


Table at $\mathcal{V}$:

A	A.2		2
B	B.1		4
C	A.2	$\mathcal{R}$	3
D	A.2	$\mathcal{R}$	6
E	A.2	$\mathcal{R}$	5
F	A.2	$\mathcal{R}$	6

Table at $\mathcal{R}$:

A	A.1		2
B	A.1	$\mathcal{V}$	6
C	C.3		1
D	C.3	$\mathcal{T}$	4
E	C.3	$\mathcal{U}$	3
F	C.3	$\mathcal{U}$	4

Table at $\mathcal{S}$:

A	D.2	$\mathcal{T}$	6
B	B.2		4
C	D.2	$\mathcal{T}$	4
D	D.2		3
E	D.2	$\mathcal{T}$	6
F	D.2	$\mathcal{T}$	7

Table at $\mathcal{T}$:

A	C.2	$\mathcal{R}$	3
B	D.1	$\mathcal{S}$	7
C	C.2		1
D	D.1		3
E	C.2	$\mathcal{U}$	3
F	C.2	$\mathcal{U}$	4

Table at $\mathcal{U}$:

A	C.1	$\mathcal{R}$	3
B	C.1	$\mathcal{T}$	8
C	C.1		1
D	C.1	$\mathcal{T}$	4
E	E.1		2
F	F.1		3

Distance-Vector Routing

Observations and drawbacks

- Routing records are **discovered and refined iteratively**
  - Sooner or later the whole system converges
    - **Good news spreads relatively fast**
    - Unfortunately, **bad news spreads too slow**…
- **Calculation** of routing paths as such is **distributed**
  - **Reality is perceived from the perspective of the neighbors**
    - I.e., we are relying on the information received from others
    - We cannot determine its validity
  - **If someone makes a mistake, it confuses everyone else**
    - $\Rightarrow$ **Routing by Rumor**
- **Volume of update information** is too large
  - I.e., frequency of messages with routing tables and their size
    - The whole approach is hence **not suitable for larger networks**

Distance-Vector Routing

Count-to-infinity: possible **solutions**

- **Small Infinity**
  - Space of permitted values of costs is always limited
    - This limit (i.e., as if infinity) **can be made small enough**
  - However, **choice of infinity is a tradeoff between network size and speed of convergence**
    - If it is not high enough, longer routes cannot be handled
- **Split Horizon**
  - **Routes are not advertised to nodes they were learned from**
- **Poisoned Reverse**
  - **Such routes are advertised, but their cost is set to infinity**
- **Triggered Updates**
  - Updates are sent **immediately after any change is detected**

Distance-Vector Routing

Routing Information Protocol (RIP)

- Very old protocol (in BSD UNIX since 1980s)
- Features
  - **Metric** is based on a **hop count**
    - **Infinity is 16** ⇒ routes longer than 15 hops will get unreachable
  - Routing table only supports **25 routing records**
  - **Updates are sent every 30 seconds**
    - Neighbor is considered as unavailable if update is not received within 180 seconds
  - Integrated directly into OS (daemon `routed`)
    - Runs at `L7` and uses UDP datagrams at port 520
- Disadvantages
  - Does not scale well, not stable enough, count-to-infinity, …
    - ⇒ **cannot be used in larger networks**

Link-State Routing

Link-State **Routing**

- Each router has <u>**complete**</u> **information** about the topology
- Principles
  - Reachability **status of neighbors is regularly monitored**
  - Whenever a change is detected…
    - **Update message is sent to <u>all</u> nodes**
- Features
  - Calculation is not incremental and distributed
  - Mistakes cannot influence others
  - Faster convergence, **lower overhead** and better scalability
  - But **still not suitable for larger networks**
- Example
  - OSPF (**Open Shortest Path First**)

Routing Challenges

Size of routing tables

- The larger the system, the higher the **number of records**
- Two strategies
  - **Aggregation** of routing records (if possible)
    - The same interface / neighboring gateway
    - **Shared and aligned address prefix**
  - **Default route**

Volume of update information

- Routing tables need to be interchanged in regular intervals
  - Both in case of distance-vector and link-state approaches
- Even bigger problem than the size of routing tables…

⇒ the only solution is **decomposition**

Hierarchical Routing

Routing domains

- System of networks is **decomposed into smaller parts**
  - So called routing domains in general
  - **Autonomous systems** in case of the Internet
    - Typically (but not necessarily) one ISP means one AS
- **Routing information becomes localized**
  - Different approaches are used within / across domains

Hierarchical **routing**

- **Interior** gateway protocols
  - E.g.: RIP, OSPF, …
- **Exterior** gateway protocols
  - **Path-Vector Routing**: based on reachability, not lowest costs
  - E.g.: Border Gateway Protocol (**BGP**)

Lecture Conclusion

Routing strategies

- Non-adaptive / adaptive
- Centralized / distributed
- Isolated / non-isolated

Particular approaches

- **Fixed Directory Routing**, Random Walk, **Flooding**
- **Backward Learning**, Source Routing, Hot Potato Routing
- **Distance-Vector** / Link-State / Path-Vector Routing