Advanced Computer Networking (ACN)

Exercise 3 – Solution

Prof. Dr.-Ing. Georg Carle

Sebastian Gallenmüller, Max Helm, Benedikt Jaeger,
Marcel Kempf, Patrick Sattler, Johannes Zirngibl

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Outline

Announcements

For questions and problems:

- Always use this mail address: `acn@net.in.tum.de`

Deadline first version:

- The deadline for the first version of Tutorial 3 has passed 15 minutes ago
- Commit and push your solution, if you haven't already

Next week:

- Tuesday:
  - Lecture, project & tutorials offer enough content for 5 ECTS
  - For organizatorial reasons there will be no lecture on Tuesday
- Thursday: Dies academicus - No lecture
- Second deadline is on **Thursday 07.12 14:00**!

Announcements
Guest Lecture on Tuesday, Dec 19

- Presenter: Dr. Cornelius Diekmann
  - 2017 PhD: "Formal Verification of Network Security Management"
  - since 2017: Working for Google as Site Reliability Engineer (SRE)
- Topic: The Art of SLOs
  - Service Level Objectives (SLO) are used to define reliability goals for services
  - Experience how to define and achieve realistic SLOs for services
- Lecture will be streamed
- Lecture will NOT be recorded

Problem 1: Autonomous Systems

1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
    remove all nodes in G of degree $\leq d$;
    if nothing changed **then**
        return G
    end
end

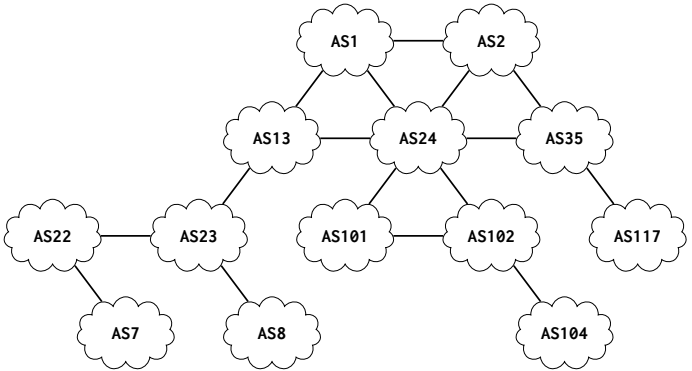
Tutorial3 – Problem 1: Autonomous Systems

1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
  remove all nodes in G of degree $\leq d$;
  if nothing changed **then**
    return G
  end
end

Example: remove nodes of degree 1



Tutorial3 – Problem 1: Autonomous Systems

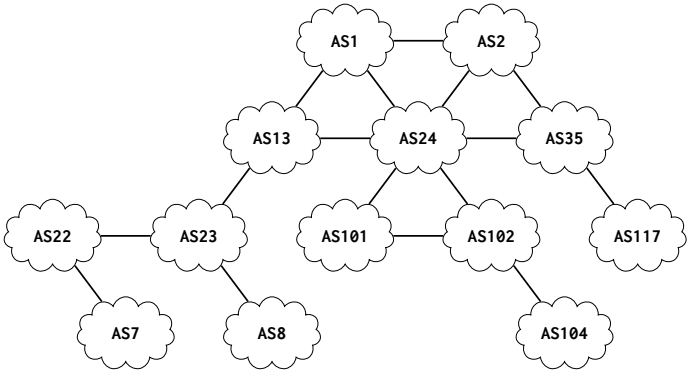
1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
    remove all nodes in G of degree $\leq d$;
    if nothing changed **then**
        return G
    end
end

Example: remove nodes of degree 1

1. Remove AS7, AS8, AS104, AS117



Tutorial3 – Problem 1: Autonomous Systems

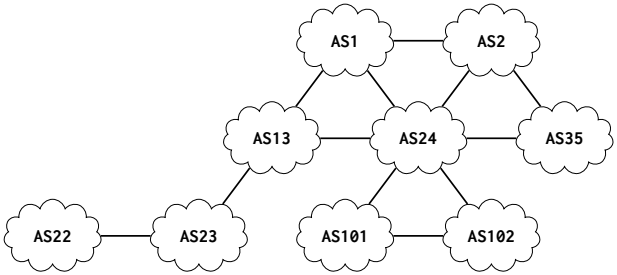
1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
  remove all nodes in G of degree $\leq d$;
  if nothing changed **then**
    return G
  end
end

Example: remove nodes of degree 1

1. Remove AS7, AS8, AS104, AS117
2. Remove AS22



Tutorial3 – Problem 1: Autonomous Systems

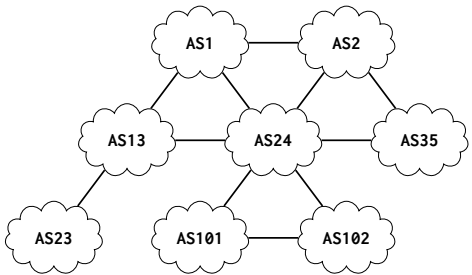
1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
  remove all nodes in G of degree $\leq d$;
  if nothing changed **then**
    return G
  end
end

Example: remove nodes of degree 1

1. Remove AS7, AS8, AS104, AS117
2. Remove AS22
3. Remove AS23



Tutorial3 – Problem 1: Autonomous Systems

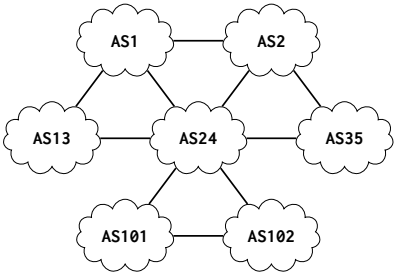
1 a)

Program a function which takes a unidirectional NetworkX graph and removes all nodes of the given or smaller degree recursively.

Input: graph G, degree d
Output: graph without nodes of degree $\leq d$
while true **do**
  remove all nodes in G of degree $\leq d$;
  if nothing changed **then**
    return G
  end
end

Example: remove nodes of degree 1

1. Remove AS7, AS8, AS104, AS117
2. Remove AS22
3. Remove AS23
4. Done



1 b)

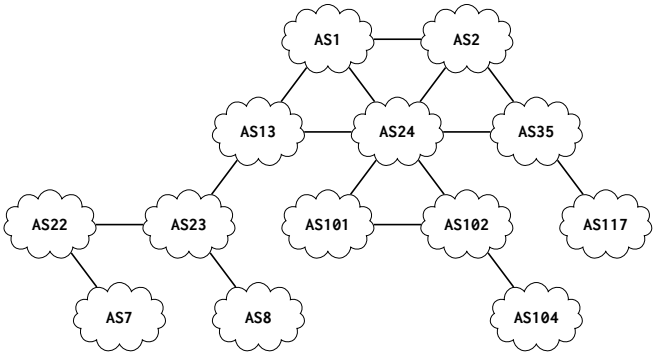
Write your own function `perform_kcore` which performs the kcore algorithm on a given unidirectional graph.

Tutorial3 – Problem 1: Autonomous Systems

1 b)

Write your own function `perform_kcore` which performs the kcore algorithm on a given unidirectional graph.

Input: graph G
Output: core of G
last $\leftarrow$ G;
core $\leftarrow$ 1;
while true **do**
    remove nodes in G of degree core;
    if G is empty **then**
        return last, core
    end
    last $\leftarrow$ G;
    core $\leftarrow$ core + 1;
end



Example:

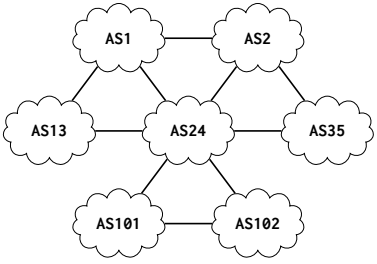
- Remove nodes of degree 1

Tutorial3 – Problem 1: Autonomous Systems

1 b)

Write your own function `perform_kcore` which performs the kcore algorithm on a given unidirectional graph.

Input: graph *G*
Output: core of *G*
last ← *G*;
core ← 1;
while true **do**
    remove nodes in *G* of degree core;
    if *G* is empty **then**
        return last, core
    end
    last ← *G*;
    core ← core + 1;
end



Example:

- Remove nodes of degree 1
- Remove nodes of degree 2

Tutorial3 – Problem 1: Autonomous Systems

1 b)

Write your own function `perform_kcore` which performs the kcore algorithm on a given unidirectional graph.

Input: graph G
Output: core of G
last $\leftarrow$ G;
core $\leftarrow$ 1;
while true **do**
    remove nodes in G of degree core;
    if G is empty **then**
        return last, core
    end
    last $\leftarrow$ G;
    core $\leftarrow$ core + 1;
end

Example:

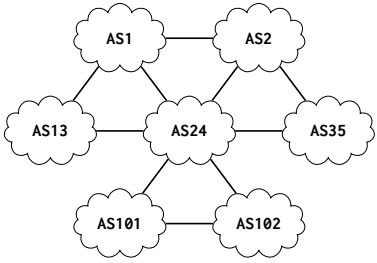
- Remove nodes of degree 1
- Remove nodes of degree 2
- Graph is empty

Tutorial3 – Problem 1: Autonomous Systems

1 b)

Write your own function `perform_kcore` which performs the kcore algorithm on a given unidirectional graph.

Input: graph G
Output: core of G
last $\leftarrow$ G;
core $\leftarrow$ 1;
while true **do**
    remove nodes in G of degree core;
    if G is empty **then**
        return last, core
    end
    last $\leftarrow$ G;
    core $\leftarrow$ core + 1;
end



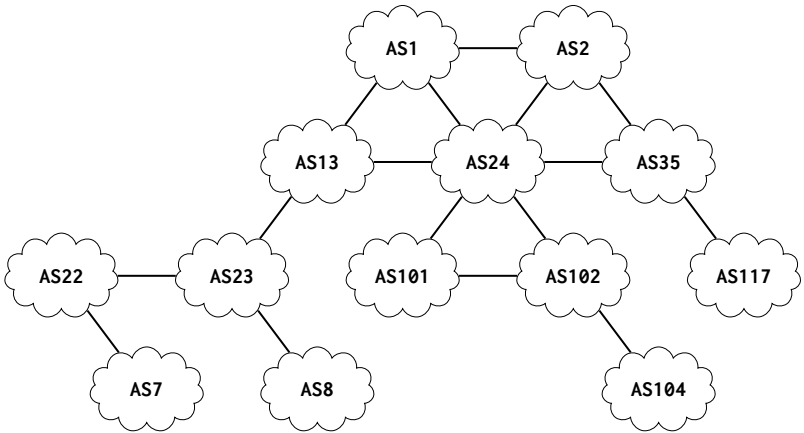
Example:

- Remove nodes of degree 1
- Remove nodes of degree 2
- Graph is empty
- Core is 2

Tutorial3 – Problem 1: Autonomous Systems

1 c)

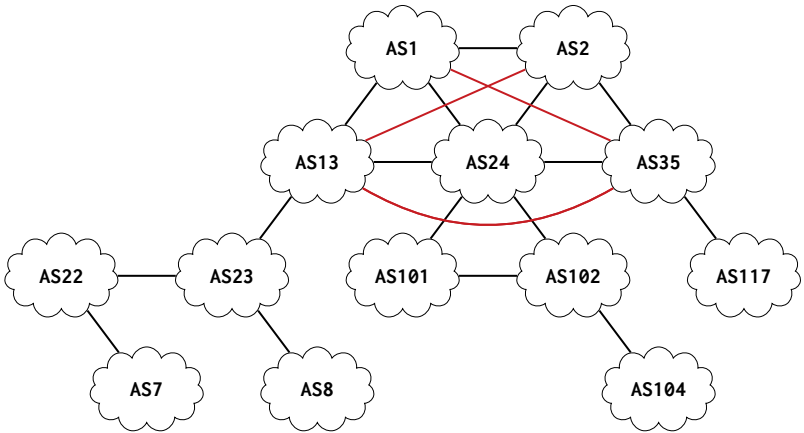
Increase the degree of the nodes in the core of the network for the given topology.
You may add exactly one connection between two arbitrary ASes.



Tutorial3 – Problem 1: Autonomous Systems

1 c)

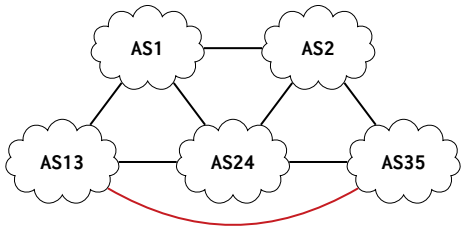
Increase the degree of the nodes in the core of the network for the given topology.
You may add exactly one connection between two arbitrary ASes.



Tutorial3 – Problem 1: Autonomous Systems

1 c)

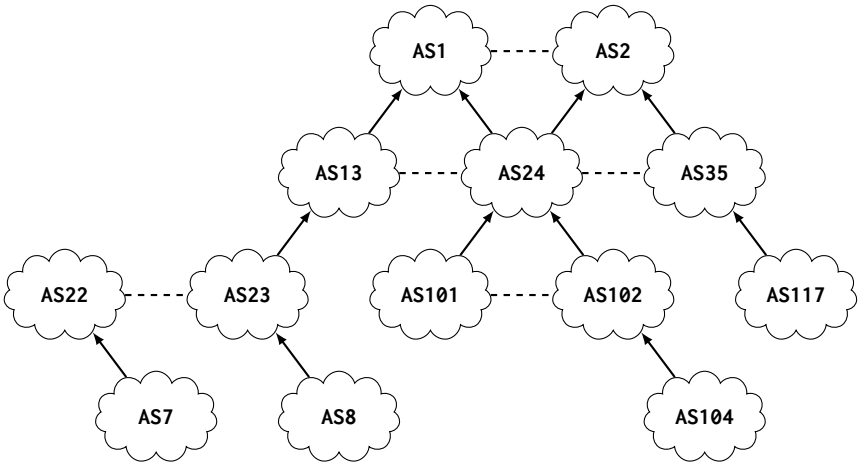
Increase the degree of the nodes in the core of the network for the given topology.
You may add exactly one connection between two arbitrary ASes.



Tutorial3 – Problem 1: Autonomous Systems

1 d and e)

Define the terms "Tier-1 provider", "stub network", "multi-homed AS", "peering" and "transit".

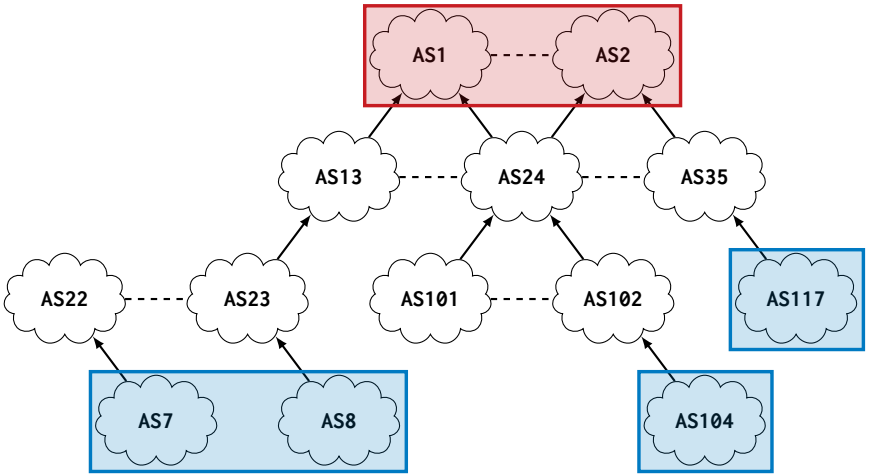


Tutorial3 – Problem 1: Autonomous Systems

1 d and e)

Define the terms "Tier-1 provider", "stub network", "multi-homed AS", "peering" and "transit".

Relevant RFCs: e.g. 7454, 1772, 4384



1 f)

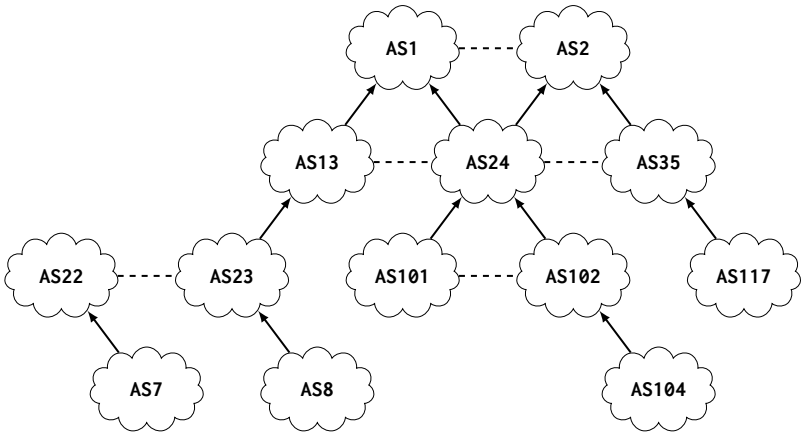
How can a transit agreement be trivially represented in the routing table of the client?

- As client, you expect your provider to offer connectivity to the Internet
- If you do not have a better/cheaper route you want to use the provider

→ You can use a default route

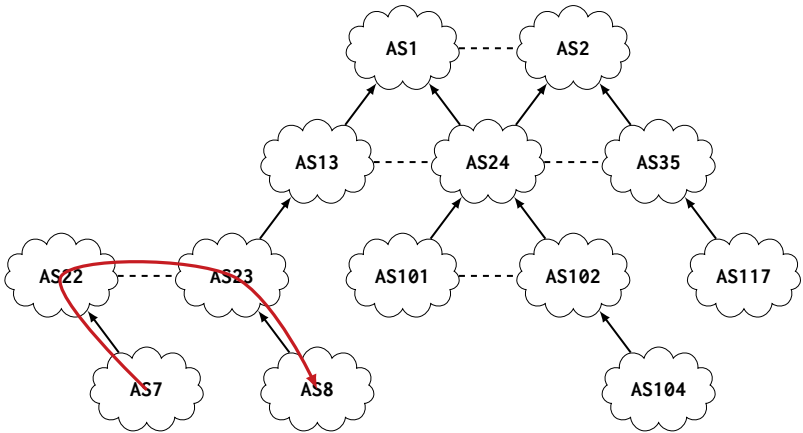
1 g)

Reason how the traffic to AS8 will be routed on AS level for sources in AS7, AS101, AS104 and AS117 considering common peering and transit agreements?



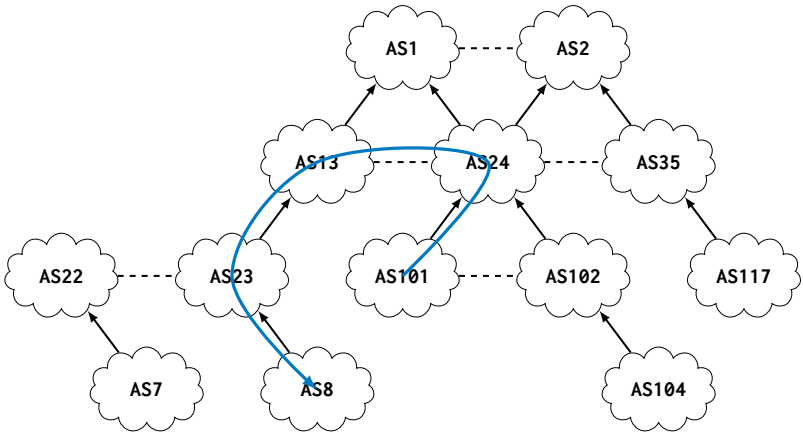
1 g)

Reason how the traffic to AS8 will be routed on AS level for sources in AS7, AS101, AS104 and AS117 considering common peering and transit agreements?



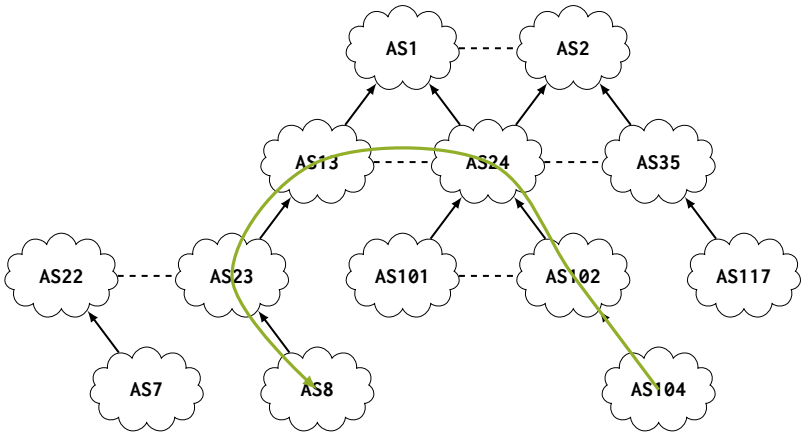
1 g)

Reason how the traffic to AS8 will be routed on AS level for sources in AS7, AS101, AS104 and AS117 considering common peering and transit agreements?



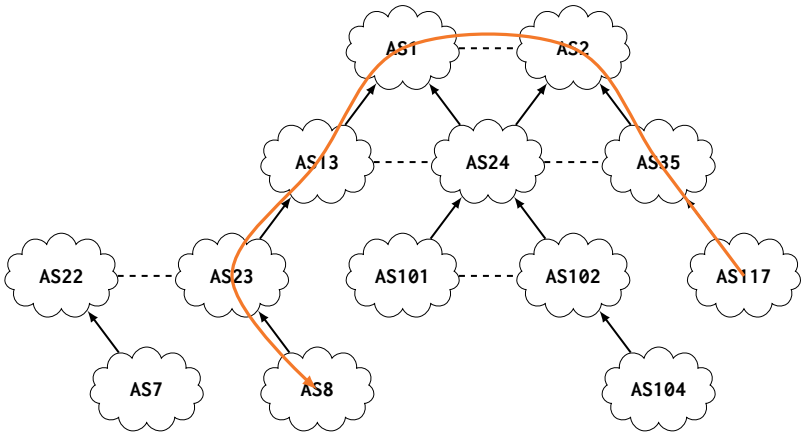
1 g)

Reason how the traffic to AS8 will be routed on AS level for sources in AS7, AS101, AS104 and AS117 considering common peering and transit agreements?



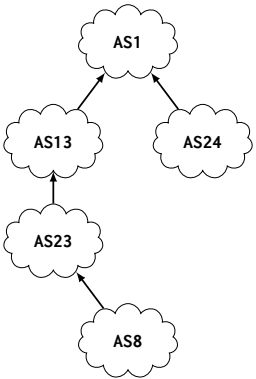
1 g)

Reason how the traffic to AS8 will be routed on AS level for sources in AS7, AS101, AS104 and AS117 considering common peering and transit agreements?



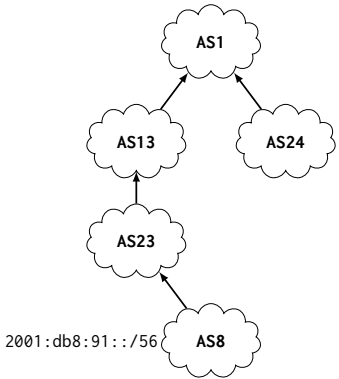
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.



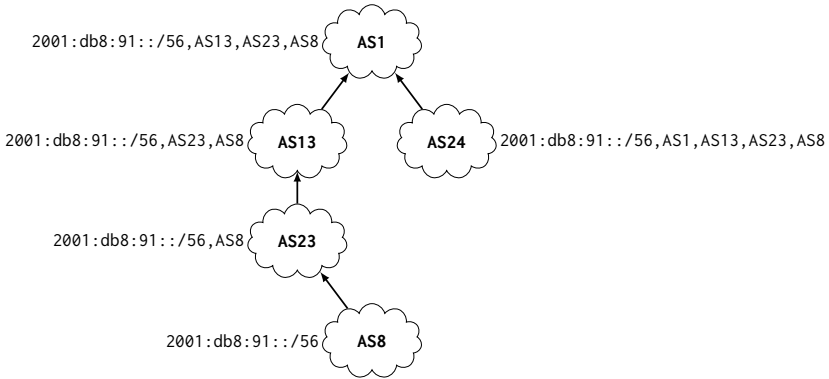
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.



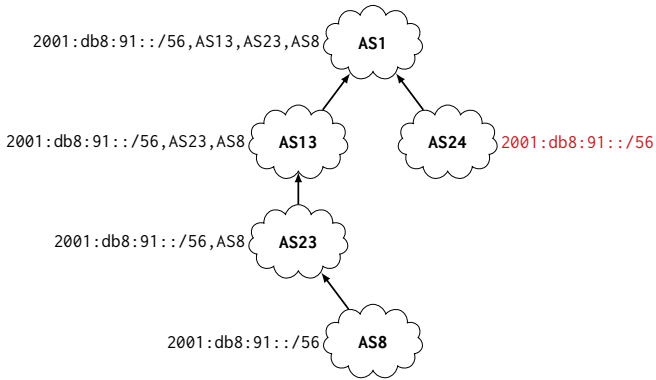
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.



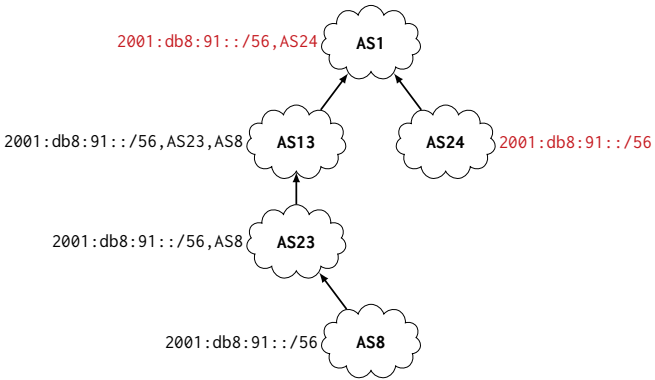
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
Prefix Hijack:



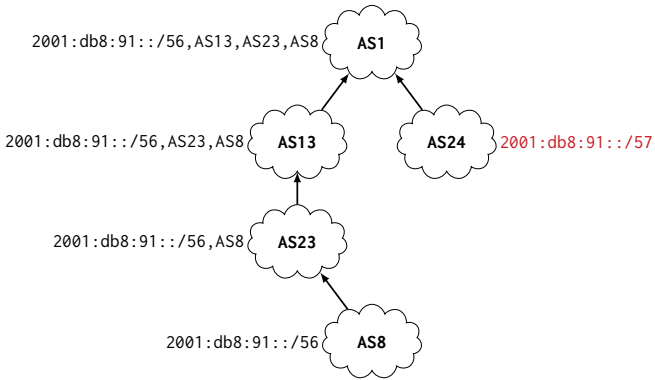
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
Prefix Hijack:



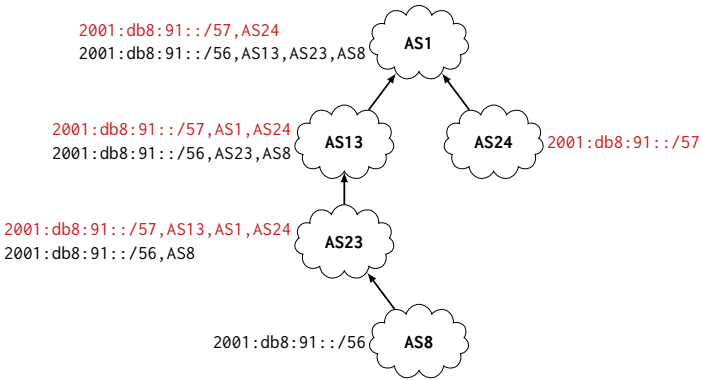
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
Subprefix Hijack:



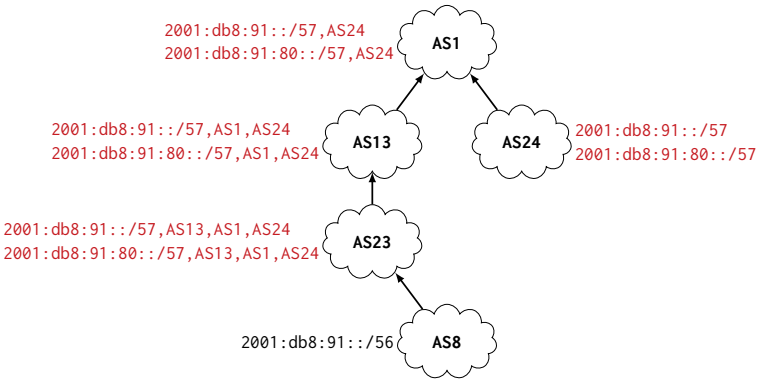
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
Subprefix Hijack:



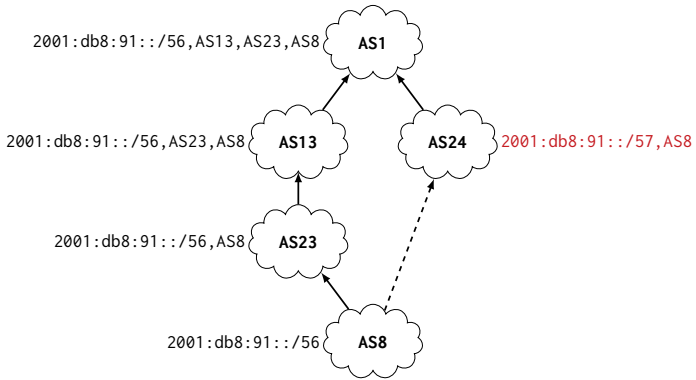
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
Subprefix Hijack:



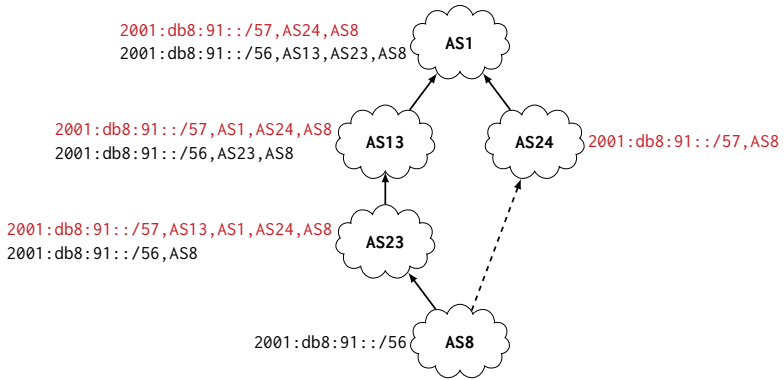
1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
(Fake) Path Hijack:



1 h-j)

BGP Hijacking is a major threat. Attacks are categorized into different classes based on their properties.
(Fake) Path Hijack:



1 k)

- Internet Routing Registries (IRR) filtering
  - Central places with public routing information
  - Used to validate announcements
- Route Origin Validation via RPKI
  - Signatures of prefix announcements from the owning AS at RIR — Route Origin Announcements (ROAs)
  - RIRs are trust anchor and distribute ROAs
  - ROV means validating announcements against available ROAs
  - ROV **can not** protect from path hijacks
- Use information from provisioning process (MANRS)
  - Mutually Agreed Norms for Routing Security (MANRS) — global initiative to improve routing security