Aalto University
School of Electrical
Engineering

Basic Principles in Networking

Message Integrity and Message Authentication Codes

Stephan Sigg

Department of Communications and Networking
Aalto University, School of Electrical Engineering
stephan.sigg@aalto.fi

Version 1.0

Aalto University
School of Electrical
Engineering

Motivation (5 min)

Modulo arithmetic revisited

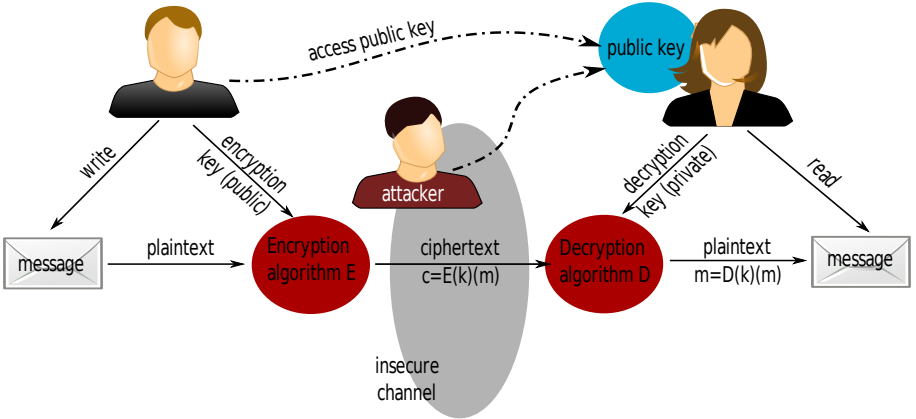
Aalto University
School of Electrical
Engineering

Part I (10 min)

Asymmetric Cryptography

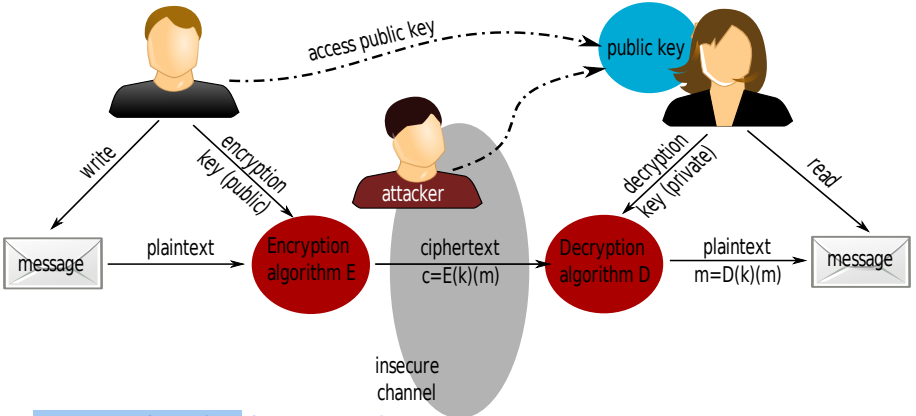
Asymmetric Cryptography

Asymmetric Encryption



Asymmetric Cryptography

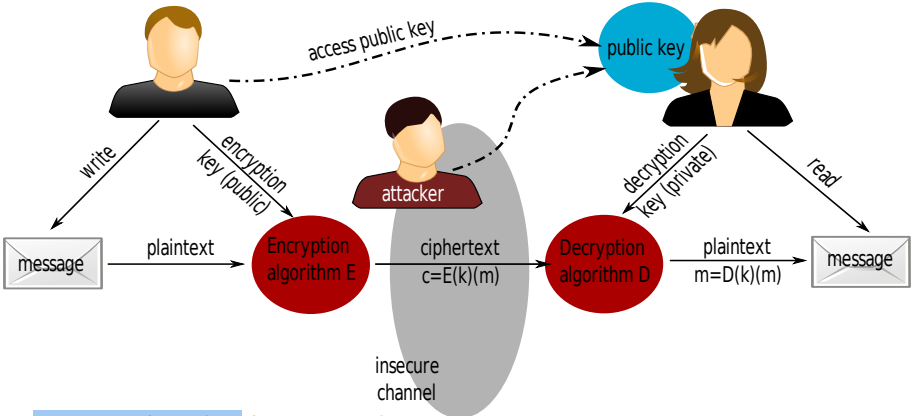
Asymmetric Encryption



Requires a one-way function for encryption

Asymmetric Cryptography

Asymmetric Encryption



Requires a one-way function for encryption

Authentication required since anybody can send encrypted messages

Asymmetric encryption

Example

RSA key generation

1. Select two large random prime numbers p and q
2. Compute $n = p \cdot q$
3. Compute $\theta(n) = (p-1) \cdot (q-1)$
4. Select small odd integer e that is relatively prime to $\theta(n)$

Asymmetric encryption

Example

RSA key generation

5. Compute d as the multiplicative inverse of $e \mod \theta(n)$
6. Publish $P = (e, n)$ as the RSA public key
7. Keep the secret pair $S = (d, n)$ as the RSA secret key

Asymmetric encryption

Example

RSA key generation

8. Encrypt a message M
   - $C = M^e \mod n$
9. Decrypt a message C
   - $M = C^d \mod n$

$x \mod n$ means
The remainder of x when deviced by n

Security stems from the fact that it is cheap to guess and verify (e.g. Solovay/Strassen) large prime numbers and to compute their product, but that factorization (needed for computation of d) is costly.

Asymmetric encryption

RSA key generation – example

- Select two prime numbers p and q
  - $p = 3; q = 11$
- Compute $n = p \cdot q$
  - $n = 33$
- Compute $\theta(n) = (p - 1) \cdot (q - 1)$
  - $\theta(n) = 20$

Asymmetric encryption

RSA key generation – example

$p = 3; q = 11; n = 33; \theta(n) = 20$

- Select a small odd integer e that is relatively prime to $\theta(n)$
  - $e = 3$
- Compute d as the multiplicative inverse of $e \mod \theta(n)$
  - $d = 7$
  - Test: $3 \cdot 7 \mod 20 = 21 \mod 20 = 1$

Asymmetric encryption

RSA key generation – example

$p = 3; q = 11; n = 33; \theta(n) = 20; e = 3; d = 7$

- Publish $P = (e, n)$ as the public key
  - Key pair: $(3, 33)$
- Keep $S = (d, n)$ as the RSA secret key
  - Secret key pair: $(7, 33)$

Asymmetric encryption

RSA key generation – example

$p = 3; q = 11; n = 33; \theta(n) = 20; e = 3; d = 7$

$P = (3, 33)$

$S = (7, 33)$

- Encrypt the message ’3’:
  - $C = M^e \mod n$
  - $C = 3^3 \mod 33 = 27 \mod 33 = 27$

Asymmetric encryption

RSA key generation – example

$p = 3; q = 11; n = 33; \theta(n) = 20; e = 3; d = 7$

$P = (3, 33)$

$S = (7, 33)$

- Decrypt the message $C = 27$:
  - $M = C^d \mod n$
  -

$$\begin{aligned} M &= 27^7 \mod 33 \\ &= 10460353203 \mod 33 \\ &= 3 \end{aligned}$$

Asymmetric encryption – example

RSA encryption, $e = 5$, $n = 35$

Plaintext letter	ASCII code	binary	m: numeric representation	m^e	Ciphertext $c \equiv m^e \mod n$
l	108	01101100	12	248832	17
o	111	01101111	15	759375	15
v	118	01110110	22	5153632	22
e	101	01100101	5	3125	10

Asymmetric encryption – example

RSA encryption, $e = 5$, $n = 35$

Plaintext letter	ASCII code	binary	m: numeric representation	m^e	Ciphertext $c \equiv m^e \mod n$
l	108	01101100	12	248832	17
o	111	01101111	15	759375	15
v	118	01110110	22	5153632	22
e	101	01100101	5	3125	10

RSA decryption, $d = 29$, $n = 35$

Ciphertext c	c^d	$m \equiv c^d \mod n$	plaintext letter
17	481968572106750915091411825223071697	12	l
15	12783403940394885893911123275756835937 5	15	o
22	85164331908653770195619449972 1106030592	22	v
10	10000000000000000000000000000000	5	e

Aalto University
School of Electrical
Engineering

Part II (20 min)

Introduction
Packet routing
Dijkstra

Introduction

Message Integrity problem

On receiving a message from Bob, Alice needs to verify

1. Whether the message indeed originated from Alice
2. Whether the message was not tampered with on its way through the network

Introduction

Message Integrity problem

On receiving a message from Bob, Alice needs to verify

1. Whether the message indeed originated from Alice
2. Whether the message was not tampered with on its way through the network

Message integrity is a critical concern in just about all secure networking protocols

Introduction

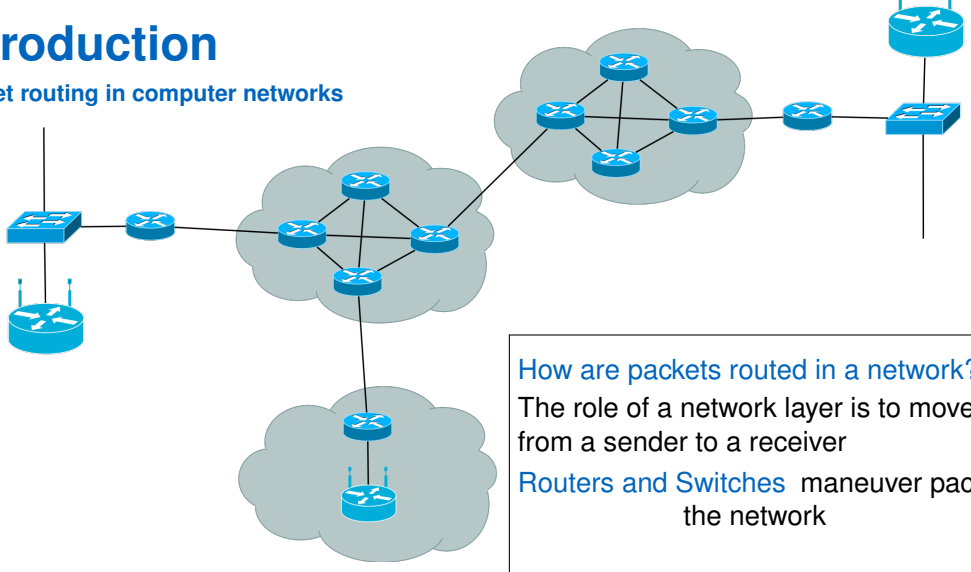
packet routing in computer networks

How are packets routed in a network?

The role of a network layer is to move packets from a sender to a receiver

Introduction

packet routing in computer networks



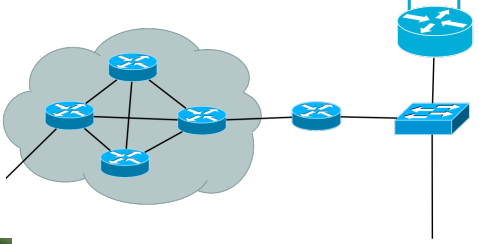
How are packets routed in a network?

The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

Introduction

packet routing in computer networks



Network Switches

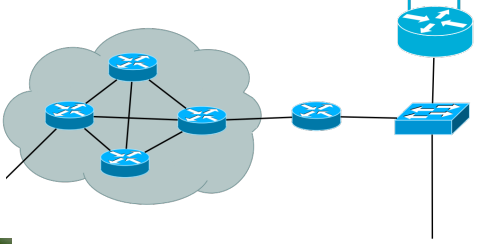
How are packets routed in a network?

The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

Introduction

packet routing in computer networks



Network Switches

How are packets routed in a network?

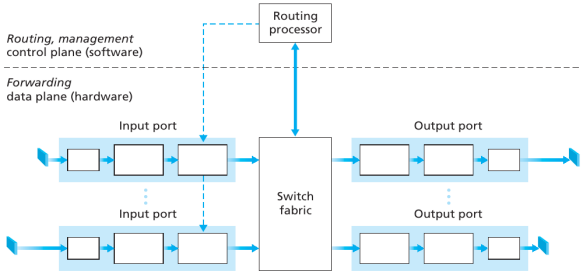
The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

How does a router work?

Introduction

packet routing in computer networks



Network Switches

How are packets routed in a network?

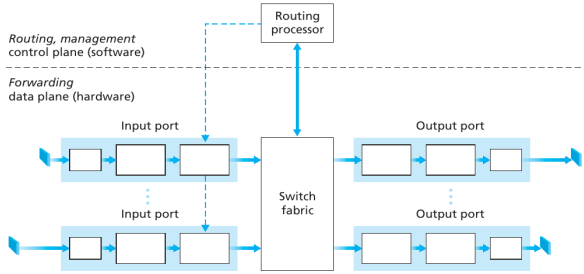
The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

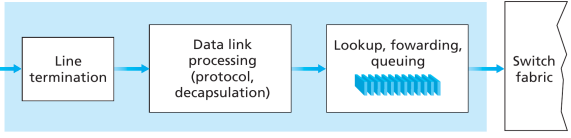
How does a router work?

Introduction

packet routing in computer networks



Input port processing



How are packets routed in a network?

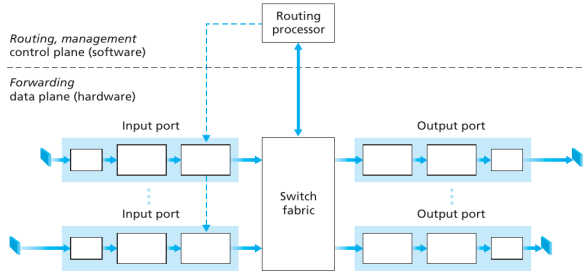
The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

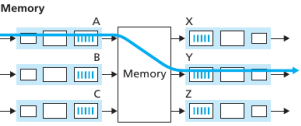
How does a router work?

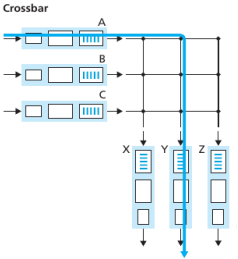
Introduction

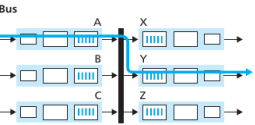
packet routing in computer networks



Three switching techniques:







How are packets routed in a network?

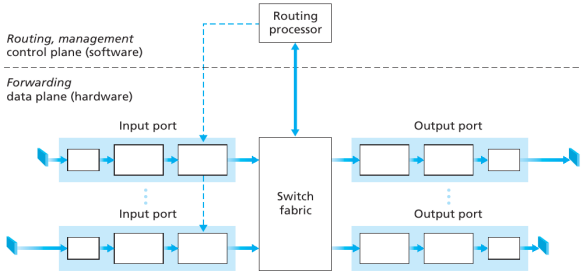
The role of a network layer is to move packets from a sender to a receiver

Routers and Switches maneuver packets in the network

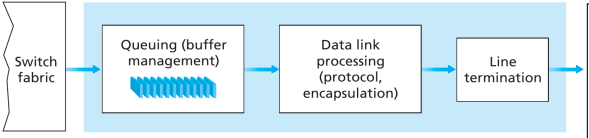
How does a router work?

Introduction

packet routing in computer networks



Output port processing:



How are packets routed in a network?

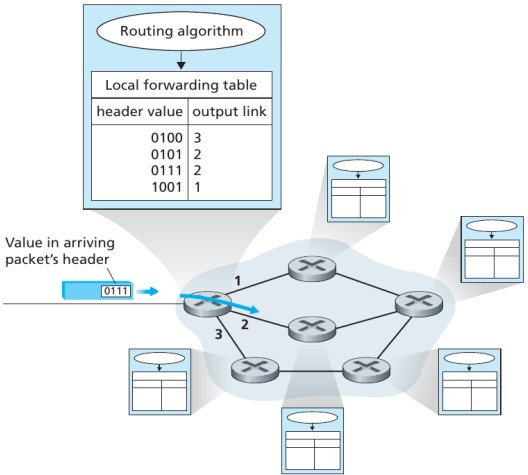
The role of a network layer is to move packets from a sender to a receiver

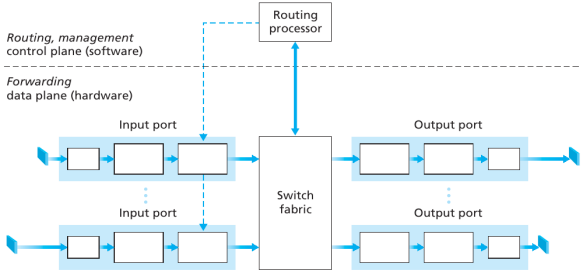
Routers and Switches maneuver packets in the network

How does a router work?

Introduction

packet routing in computer networks



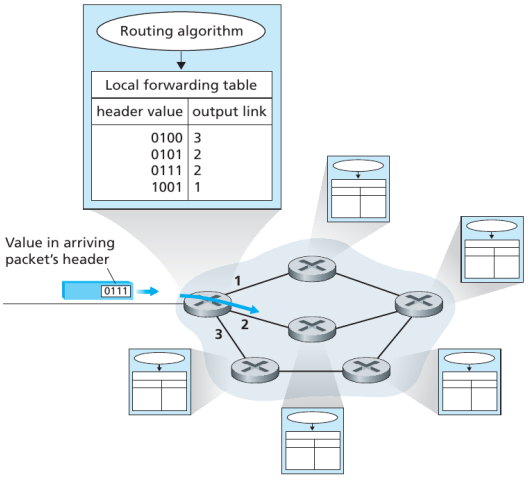


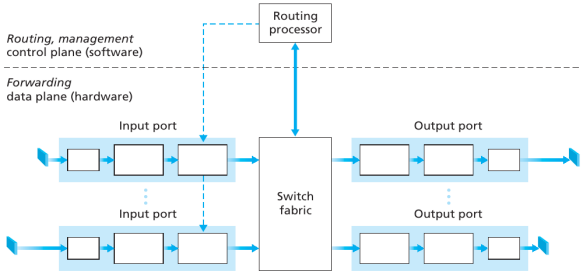
How are packets routed in a network?

The role of a network layer is to move packets from a sender to a receiver

Introduction

packet routing in computer networks





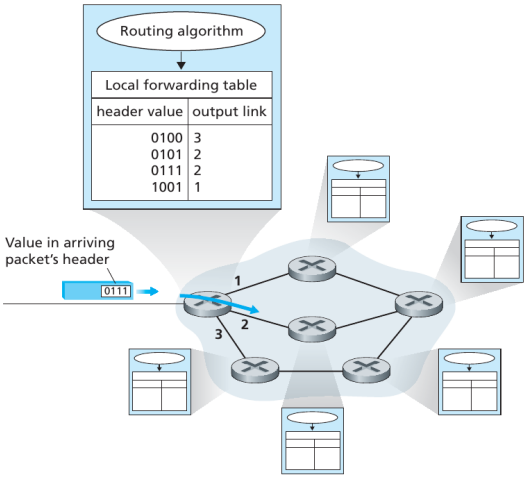
How are packets routed in a network?

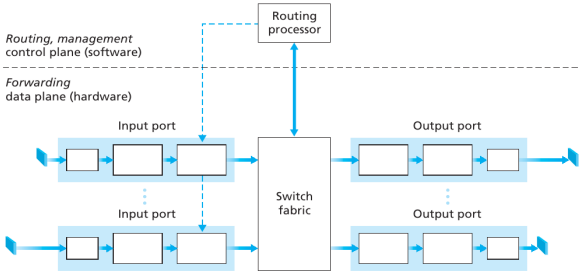
The role of a network layer is to move packets from a sender to a receiver

Forwarding When a packet arrives at a routers input link, it must be transferred to the appropriate output link

Introduction

packet routing in computer networks





How are packets routed in a network?

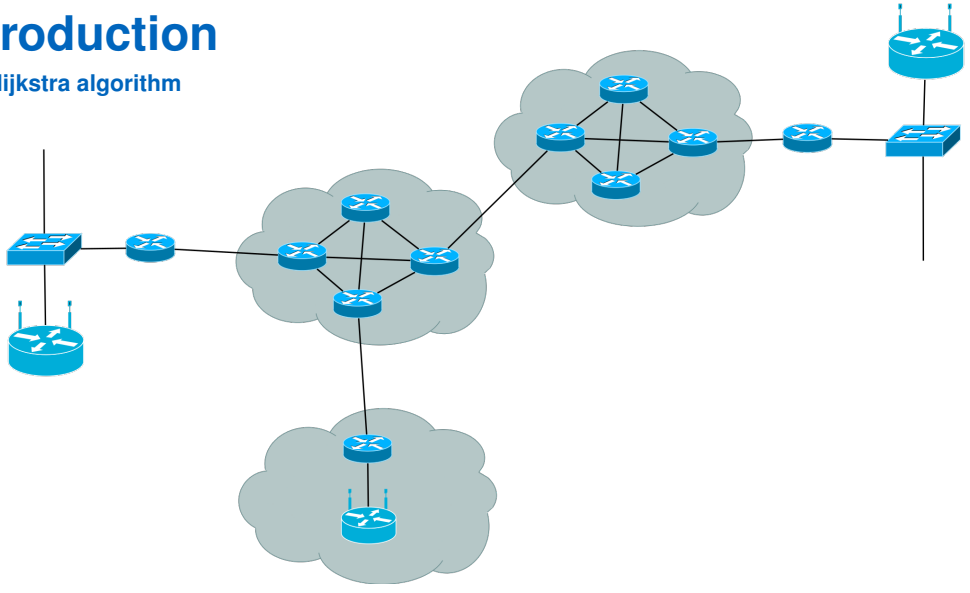
The role of a network layer is to move packets from a sender to a receiver

Forwarding When a packet arrives at a routers input link, it must be transferred to the appropriate output link

Routing Network layer must determine the route taken by packets

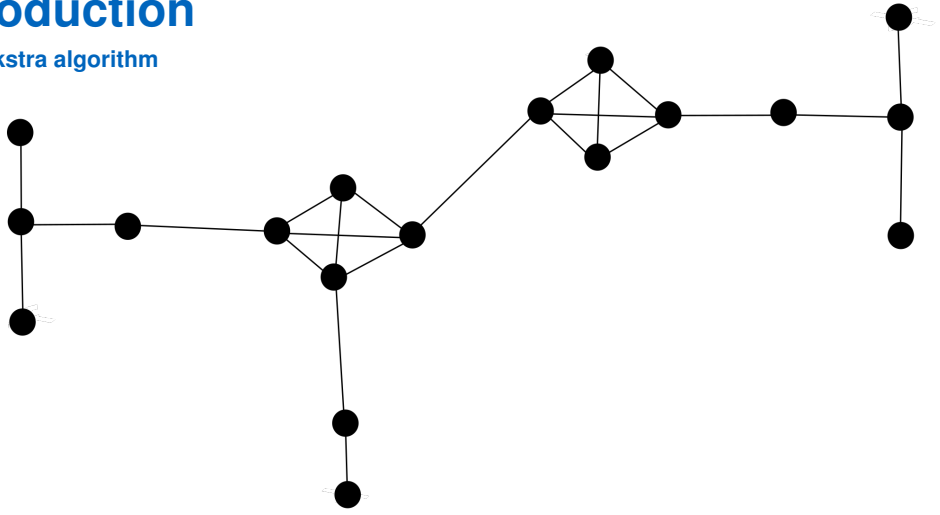
Introduction

The dijkstra algorithm



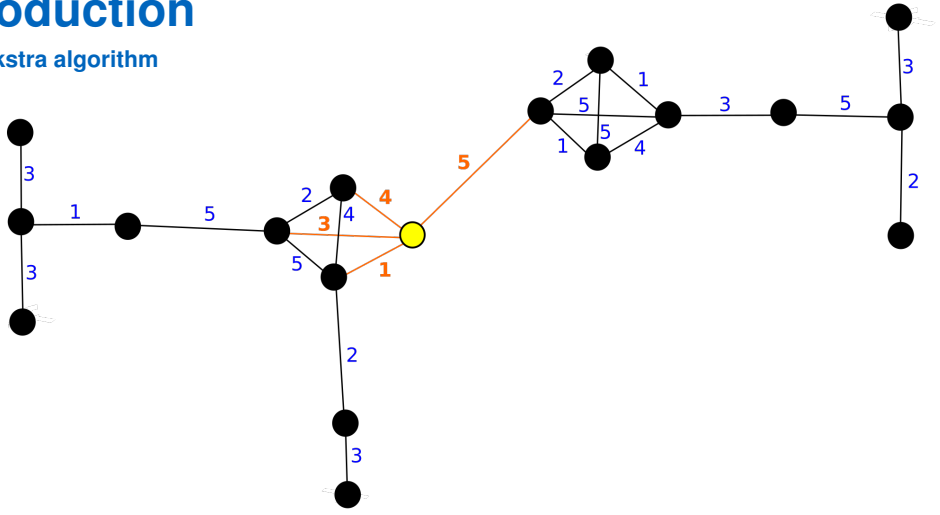
Introduction

The dijkstra algorithm



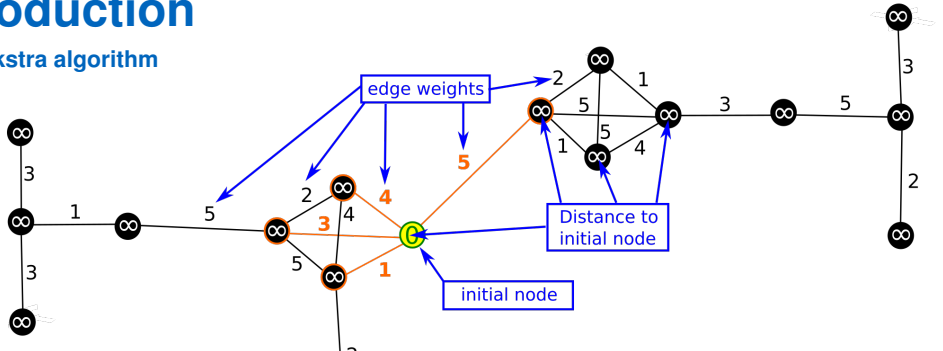
Introduction

The dijkstra algorithm



Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

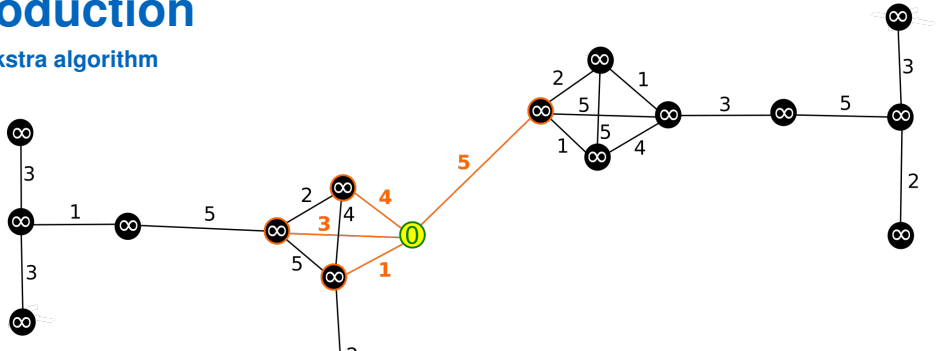
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

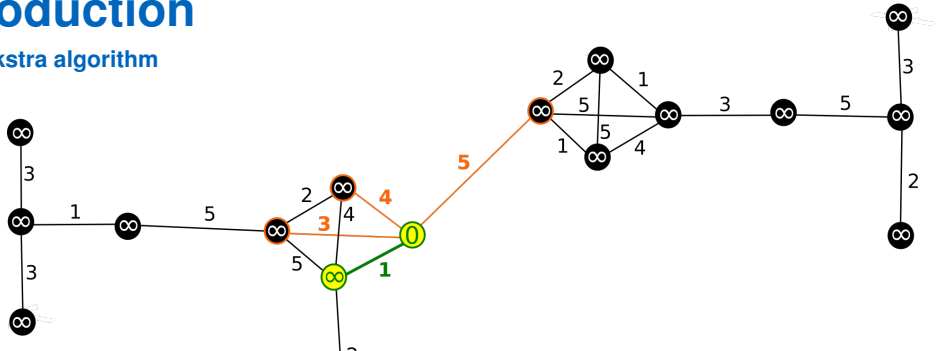
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

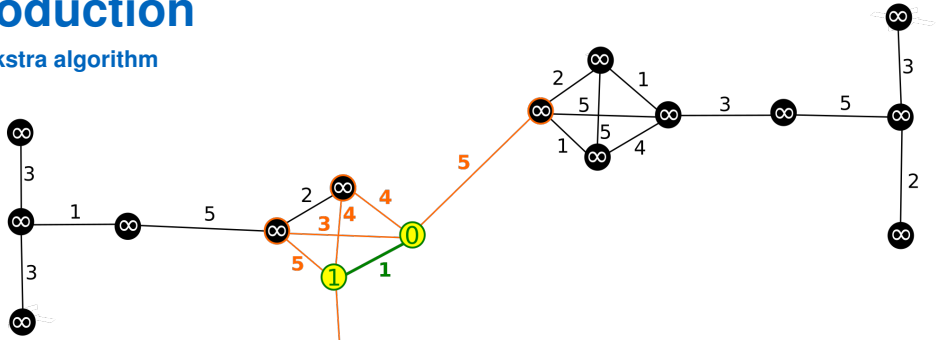
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

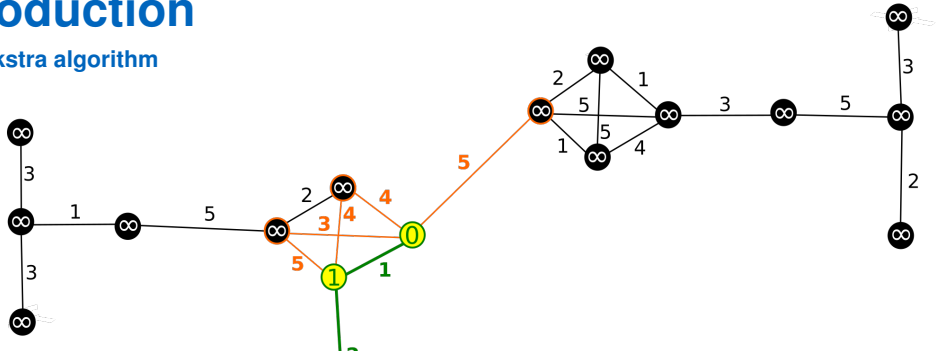
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

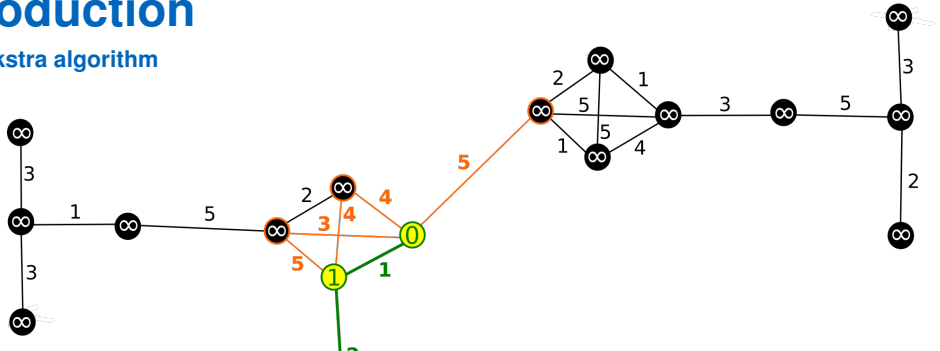
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

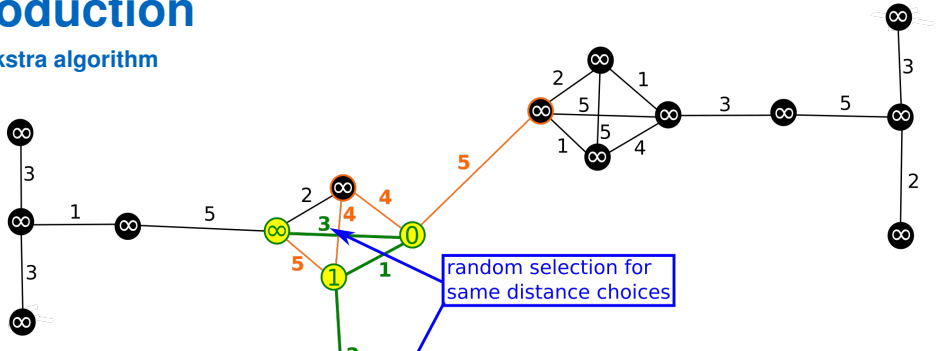
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm

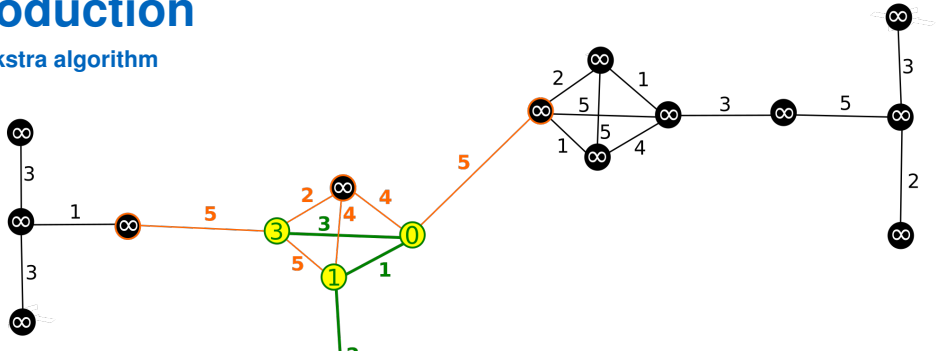


1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set
Node in the visited set
Node in the remaining set
Disregarded edge
Edge between nodes in the remaining set
Chosen edge
Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

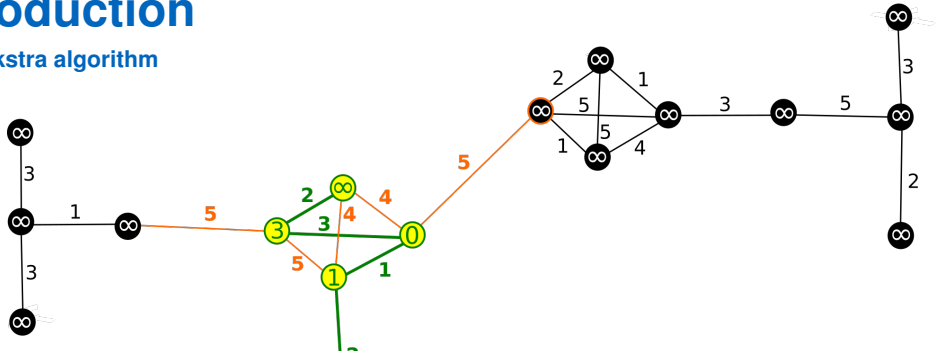
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

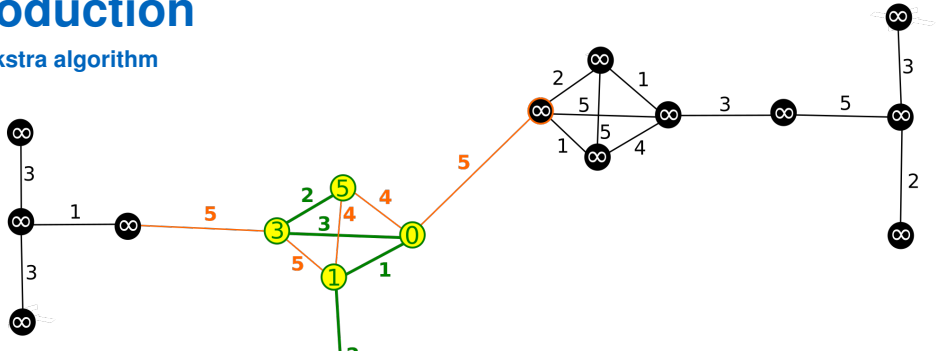
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

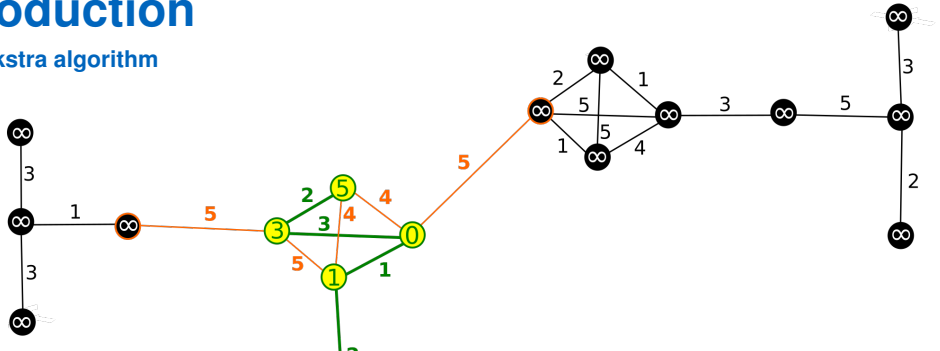
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

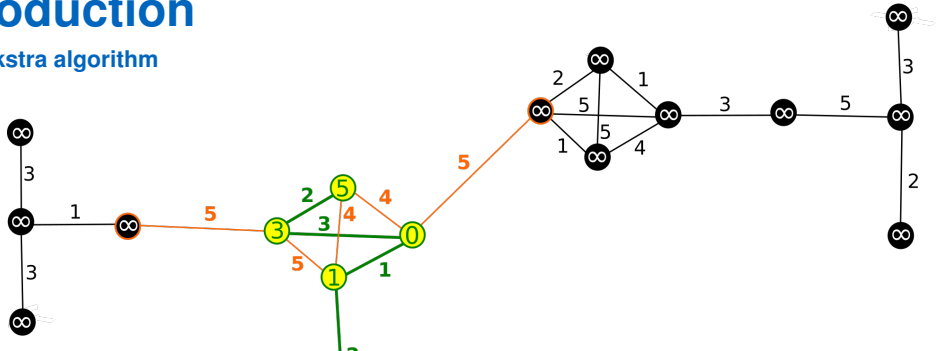
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

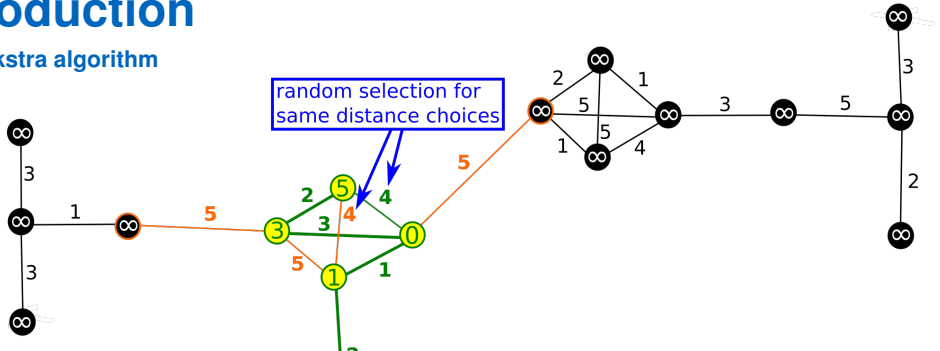
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

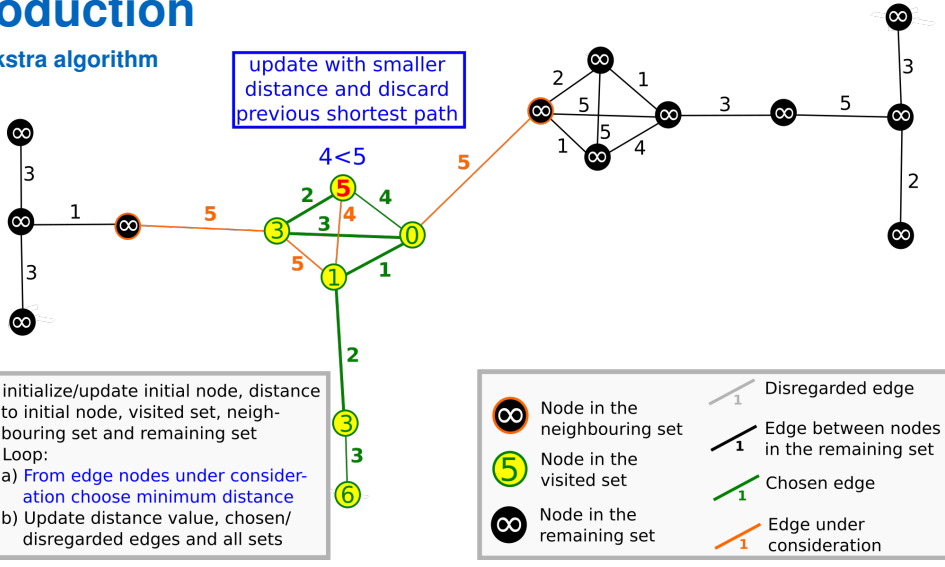
Edge between nodes in the remaining set

Chosen edge

Edge under consideration

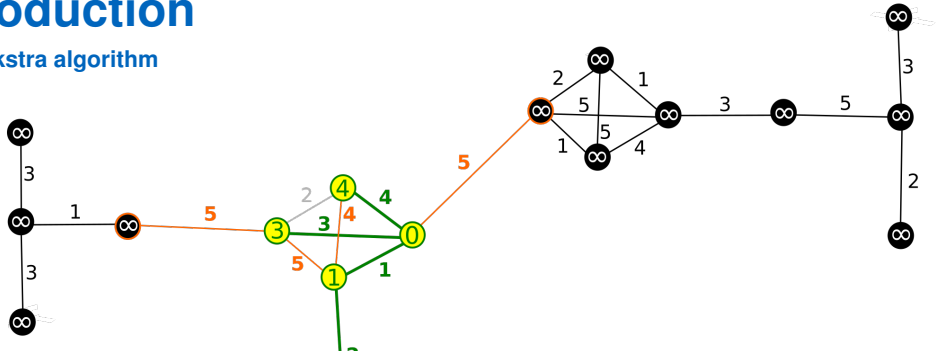
Introduction

The dijkstra algorithm



Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

∞ Node in the neighbouring set

5 Node in the visited set

∞ Node in the remaining set

1 Disregarded edge

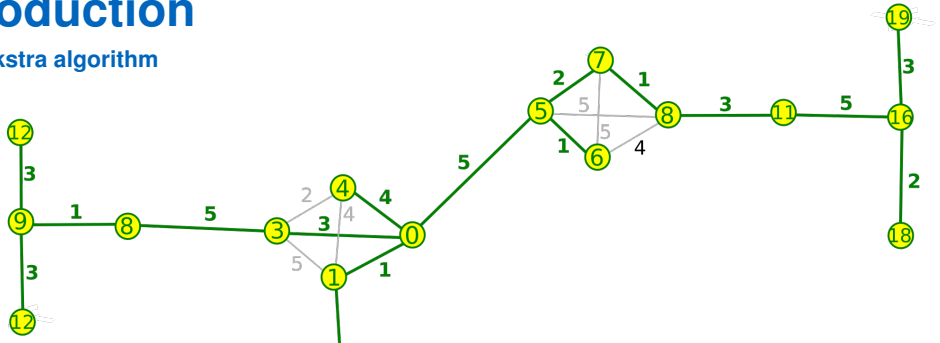
1 Edge between nodes in the remaining set

1 Chosen edge

1 Edge under consideration

Introduction

The dijkstra algorithm



1) initialize/update initial node, distance to initial node, visited set, neighbouring set and remaining set
2) Loop:
   a) From edge nodes under consideration choose minimum distance
   b) Update distance value, chosen/ disregarded edges and all sets

Node in the neighbouring set

Node in the visited set

Node in the remaining set

Disregarded edge

Edge between nodes in the remaining set

Chosen edge

Edge under consideration

Introduction

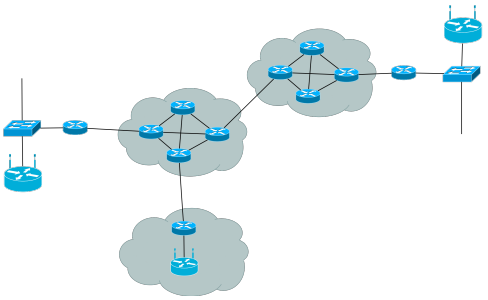
OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs

Introduction

OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs

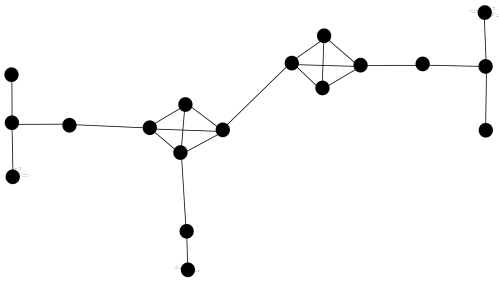


1. Routers broadcast link-state information with each link state change

Introduction

OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs

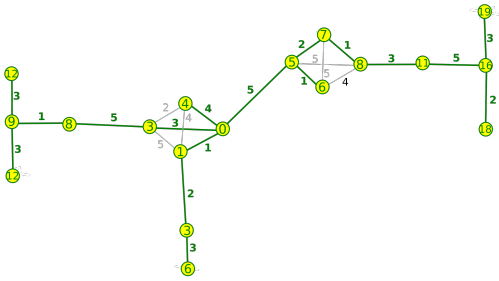


1. Routers broadcast link-state information with each link state change
2. Router constructs complete topological map (graph) of the entire AS

Introduction

OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs

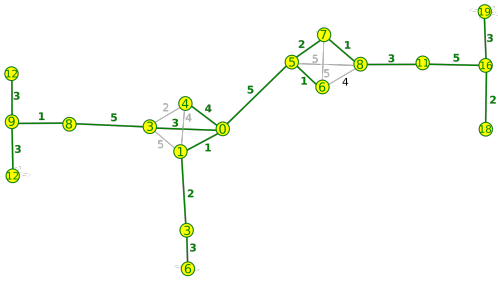


1. Routers broadcast link-state information with each link state change
2. Router constructs complete topological map (graph) of the entire AS
3. Locally runs Dijkstra to determine a shortest-path tree to all subnets

Introduction

OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs

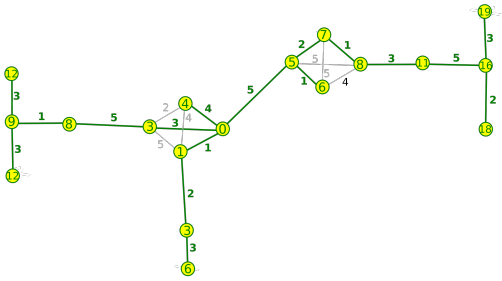


1. Routers broadcast link-state information with each link state change
2. Router constructs complete topological map (graph) of the entire AS
3. Locally runs Dijkstra to determine a shortest-path tree to all subnets
4. Individual link costs are set by admin

Introduction

OSPF: Open Shortest Path First (Example)

- For intra-AS (autonomous system) routing in Internet (RFC 2328)
- Deployed in upper-tier ISPs



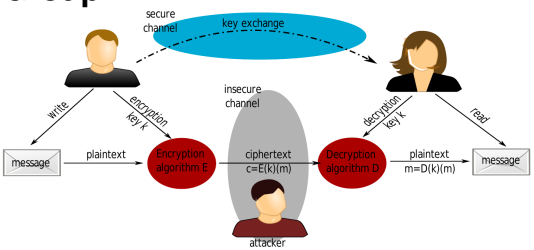
1. Routers broadcast link-state information with each link state change
2. Router constructs complete topological map (graph) of the entire AS
3. Locally runs Dijkstra to determine a shortest-path tree to all subnets
4. Individual link costs are set by admin
   - minimum-hop routing: all costs equal
   - discouraging low-bandwidth links: costs inversely proportional to link capaciy

Aalto University
School of Electrical
Engineering

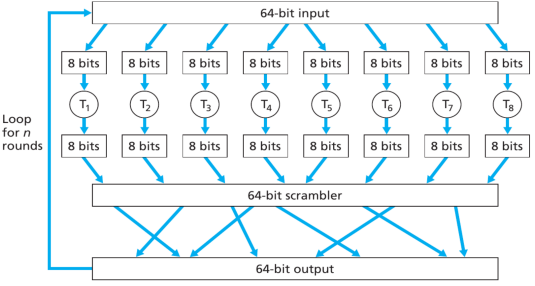
Recap-slam (15 min)

Recap-Slam

Group A:



Group B:



Preparation 5 minutes

Presentation Group A 5 minutes

Presentation Group B 5 minutes

Aalto University
School of Electrical
Engineering

Part II (20 min)

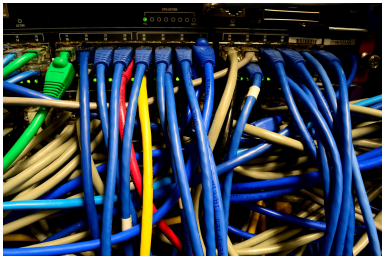
Message Integrity in OSPF
Message authentication code

Message integrity

Why message integrity is important in OSPF

In OSPF, each router broadcasts a link-state message to all other routers whenever link states change

From this information, complete maps of the network are computed by each router

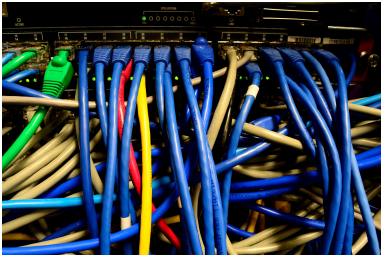


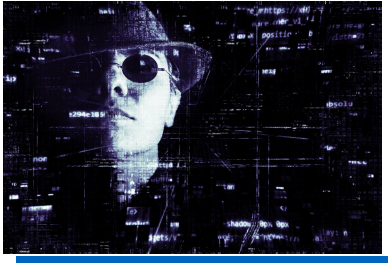
Message integrity

Why message integrity is important in OSPF

In OSPF, each router broadcasts a link-state message to all other routers whenever link states change

From this information, complete maps of the network are computed by each router





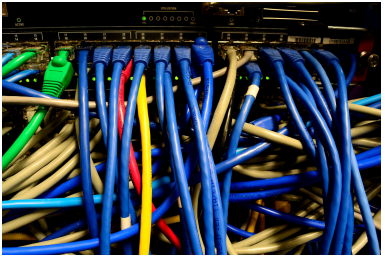
Easy attack: distribute bogus link-state messages with incorrect link-state information

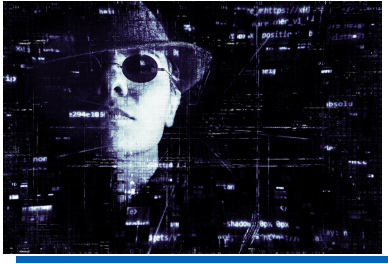
Message integrity

Why message integrity is important in OSPF

In OSPF, each router broadcasts a link-state message to all other routers whenever link states change

From this information, complete maps of the network are computed by each router





Easy attack: distribute bogus link-state messages with incorrect link-state information

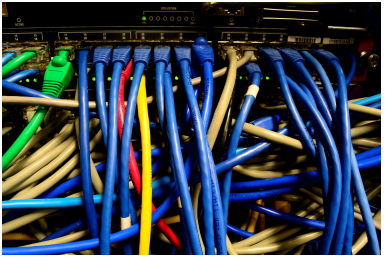
Solution: **Message integrity**: Verify sender identity and detect if messages have been tampered with

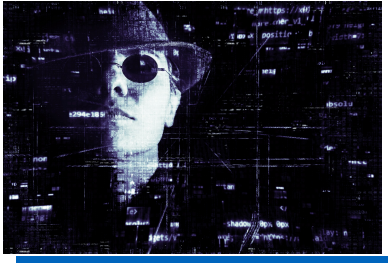
Message integrity

Why message integrity is important in OSPF

In OSPF, each router broadcasts a link-state message to all other routers whenever link states change

From this information, complete maps of the network are computed by each router





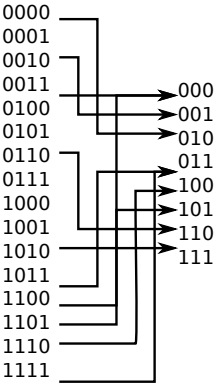
Easy attack: distribute bogus link-state messages with incorrect link-state information

Solution: **Message integrity**: Verify sender identity and detect if messages have been tampered with

Required: **Cryptographic hash functions**

Cryptographic hash functions

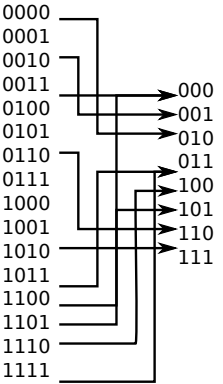
A hash function takes an input, m, and computes a hash as a fixed-size string $H(m)$.



Cryptographic hash functions

A hash function takes an input, m, and computes a hash as a fixed-size string $H(m)$.

Examples: Internet checksum, CRC

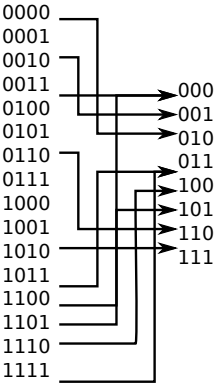


Cryptographic hash functions

A hash function takes an input, m, and computes a hash as a fixed-size string $H(m)$.

Examples: Internet checksum, CRC

For a **Cryptographic hash function** it is computationally infeasible to find any two different messages x and y such that $H(x) = H(y)$

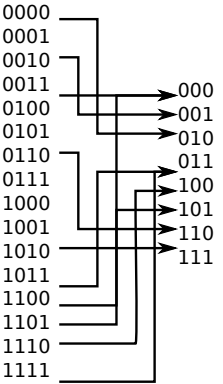


Cryptographic hash functions

A hash function takes an input, m, and computes a hash as a fixed-size string $H(m)$.

Examples: Internet checksum, CRC

For a **Cryptographic hash function** it is computationally infeasible to find any two different messages x and y such that $H(x) = H(y)$



In other words: It is infeasible to substitute another message y for x with the same hash $H(x) = H(y)$

Example: Internet checksum[1]

Message	ASCII Representation			
I O U 1	49	4F	55	31
0 0 . 9	30	30	2E	39
9 B O B	39	42	4F	42
	B2	C1	D2	AC

Checksum

0_{hex}	0_{dec}	0_{oct}
1_{hex}	1_{dec}	1_{oct}
2_{hex}	2_{dec}	2_{oct}
3_{hex}	3_{dec}	3_{oct}
4_{hex}	4_{dec}	4_{oct}
5_{hex}	5_{dec}	5_{oct}
6_{hex}	6_{dec}	6_{oct}
7_{hex}	7_{dec}	7_{oct}
8_{hex}	8_{dec}	10_{oct}
9_{hex}	9_{dec}	11_{oct}
A_{hex}	10_{dec}	12_{oct}
B_{hex}	11_{dec}	13_{oct}
C_{hex}	12_{dec}	14_{oct}
D_{hex}	13_{dec}	15_{oct}
E_{hex}	14_{dec}	16_{oct}
F_{hex}	15_{dec}	17_{oct}

[1] IOU (abbreviated from ”I owe you”): informal document acknowledging debt.

Example: Internet checksum[1]

Message	ASCII Representation			
I O U 1	49	4F	55	31
0 0 . 9	30	30	2E	39
9 B O B	39	42	4F	42
	B2	C1	D2	AC

Checksum

Message	ASCII Representation			
I O U 9	49	4F	55	39
0 0 . 1	30	30	2E	31
9 B O B	39	42	4F	42
	B2	C1	D2	AC

Checksum

0_{hex}	0_{dec}	0_{oct}
1_{hex}	1_{dec}	1_{oct}
2_{hex}	2_{dec}	2_{oct}
3_{hex}	3_{dec}	3_{oct}
4_{hex}	4_{dec}	4_{oct}
5_{hex}	5_{dec}	5_{oct}
6_{hex}	6_{dec}	6_{oct}
7_{hex}	7_{dec}	7_{oct}
8_{hex}	8_{dec}	10_{oct}
9_{hex}	9_{dec}	11_{oct}
A_{hex}	10_{dec}	12_{oct}
B_{hex}	11_{dec}	13_{oct}
C_{hex}	12_{dec}	14_{oct}
D_{hex}	13_{dec}	15_{oct}
E_{hex}	14_{dec}	16_{oct}
F_{hex}	15_{dec}	17_{oct}

[1] IOU (abbreviated from ”I owe you”): informal document acknowledging debt.

Example: Internet checksum[1]

Message	ASCII Representation			
I O U 1	49	4F	55	31
0 0 . 9	30	30	2E	39
9 B O B	39	42	4F	42
	B2	C1	D2	AC

Checksum

Message	ASCII Representation			
I O U 9	49	4F	55	39
0 0 . 1	30	30	2E	31
9 B O B	39	42	4F	42
	B2	C1	D2	AC

Checksum

Simple checksum would make poor cryptographic hash function

0_{hex}	0_{dec}	0_{oct}
1_{hex}	1_{dec}	1_{oct}
2_{hex}	2_{dec}	2_{oct}
3_{hex}	3_{dec}	3_{oct}
4_{hex}	4_{dec}	4_{oct}
5_{hex}	5_{dec}	5_{oct}
6_{hex}	6_{dec}	6_{oct}
7_{hex}	7_{dec}	7_{oct}
8_{hex}	8_{dec}	10_{oct}
9_{hex}	9_{dec}	11_{oct}
A_{hex}	10_{dec}	12_{oct}
B_{hex}	11_{dec}	13_{oct}
C_{hex}	12_{dec}	14_{oct}
D_{hex}	13_{dec}	15_{oct}
E_{hex}	14_{dec}	16_{oct}
F_{hex}	15_{dec}	17_{oct}

[1] IOU (abbreviated from ”I owe you”): informal document acknowledging debt.

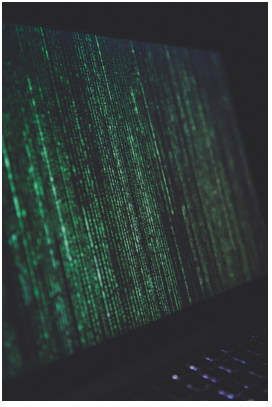
Example: Internet checksum[1]

Relevant cryptographic hash functions

MD5 hash algorithm proposed by Ron Rivest [RFC 1321] (2004: broken – not collision resistant)

SHA-2/3 US federal standard for a Secure Hash Algorithm; Required for use whentever a cryptographic hash algorithm is needed for federal applications. (2005: SHA-1 broken: chosen plaintext attack)

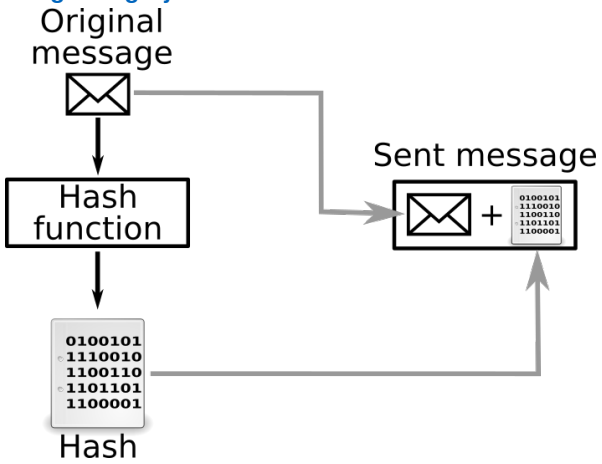
Simple checksum would make poor cryptographic hash function



[1] IOU (abbreviated from "I owe you"): informal document acknowledging debt.

Message Authentication codes

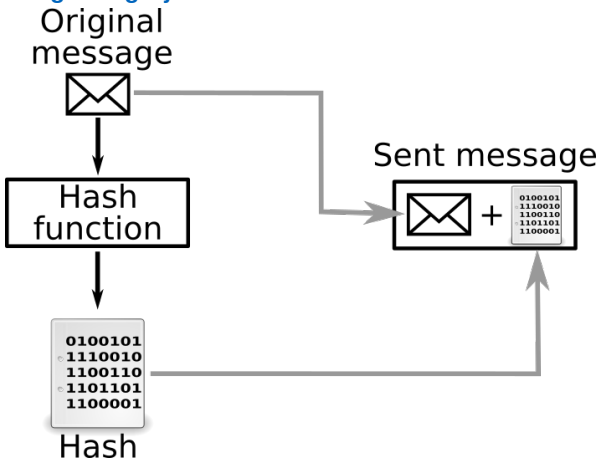
Use of hash functions to establish message integrity



Message Authentication codes

Use of hash functions to establish message integrity

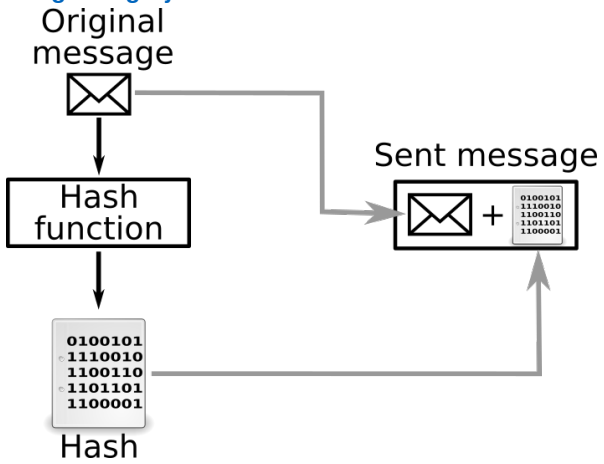
1. Create message m and calculate $H(m)$ (e.g. using SHA-2)



Message Authentication codes

Use of hash functions to establish message integrity

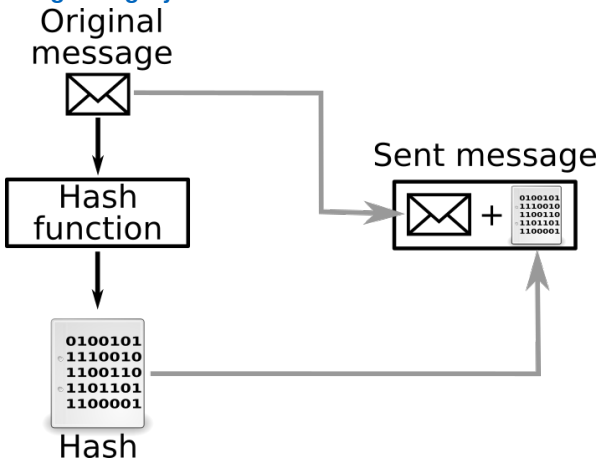
2. Create message m and calculate $H(m)$ (e.g. using SHA-2)
3. Append $H(m)$ to the message m and send $(m, H(m))$



Message Authentication codes

Use of hash functions to establish message integrity

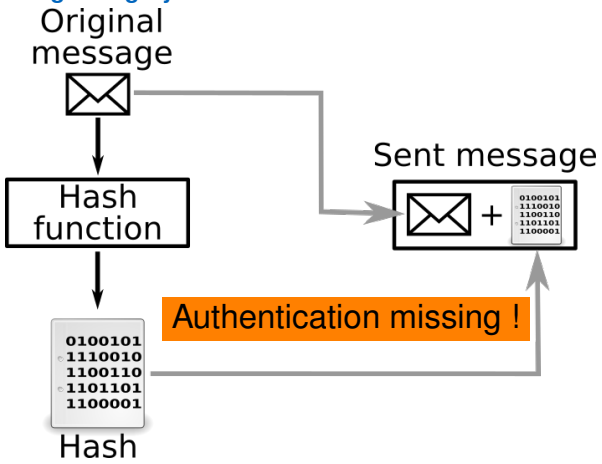
2. Create message m and calculate $H(m)$ (e.g. using SHA-2)
4. Append $H(m)$ to the message m and send $(m, H(m))$
5. Receiver of (m, h) calculates $H(m)$. Verify: $H(m) = h$



Message Authentication codes

Use of hash functions to establish message integrity

2. Create message m and calculate $H(m)$ (e.g. using SHA-2)
4. Append $H(m)$ to the message m and send $(m, H(m))$
5. Receiver of (m, h) calculates $H(m)$. Verify: $H(m) = h$



Aalto University
School of Electrical
Engineering

Video: INTERVIEW – Ron Rivest (10 min)

Some history of RSA

Questions?

Stephan Sigg
stephan.sigg@aalto.fi

Esa Vikberg
esa.vikberg@aalto.fi

Leo Lazar
leo.lazar@aalto.fi

Literature

- J.F. Kurose,K.W. Ross: Computer Networking: A Top-Down approach (7th edition), Pearson, 2016.
- J.F. Kurose,K.W. Ross: Computer Networking: A Top-Down approach (6th edition), Addison-Wesley, 2012.