# Nätverksteknik A — Introduktion till Routing

Lennart Franked

Avdelningen för informationssystem och -teknologi (IST)
Mittuniversitetet

8 oktober 2020

Innehållsförteckning

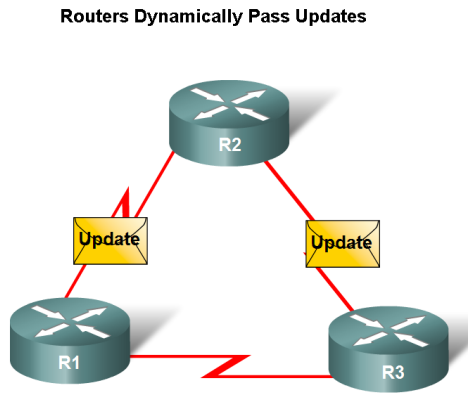
"An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process" —
Brookshear, *Computer Science: An overview*, 10th ed.

- Dynamiska routingprotokoll har funnits sedan sent 80-tal.

Tabell 1: Dynamiska routingprotokoll

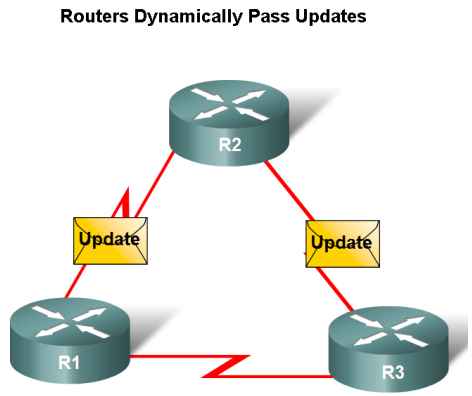
| | Interior Gateway Protocols | | | | Exterior Gateway Protocols |
|---|---|---|---|---|---|
| | Distance Vector | | Link-State | | Path Vector |
| IPv4 | RIPv2 | EIGRP | OSPFv2 | IS-IS | BGP-4 |
| IPv6 | RIPng | EIGRP for IPv6 | OSPFv3 | IS-IS for IPv6 | BGP-MP |

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina ‘interface’,
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.



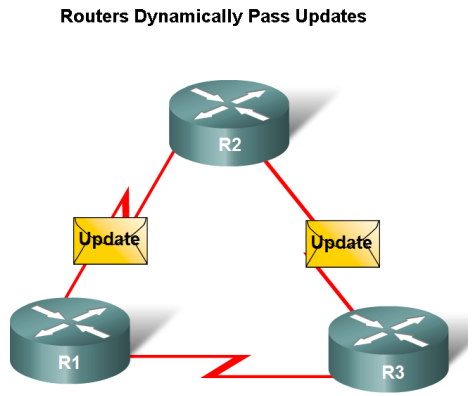

Figur 1: Dynamiskrouting[2]

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina ‘interface’,
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.



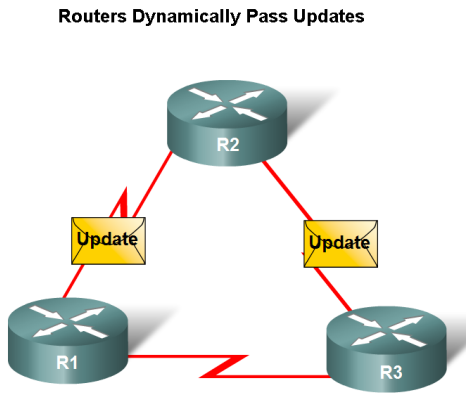

Figur 1: Dynamiskrouting[2]

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina 'interface',
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.



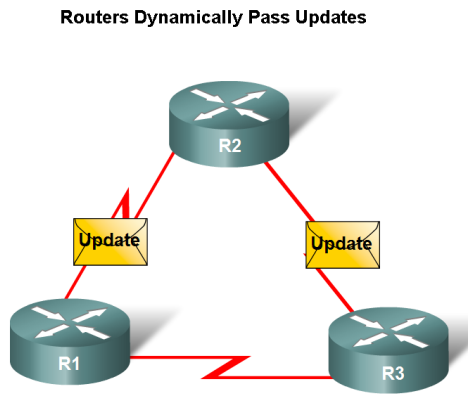

Figur 1: Dynamiskrouting[2]

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina 'interface',
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.



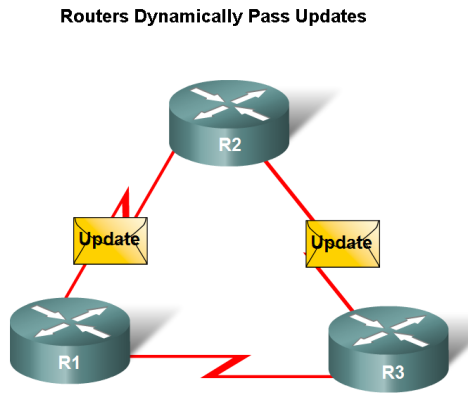

Figur 1: Dynamiskrouting[2]

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina ‘interface’,
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.




Figur 1: Dynamiskrouting[2]

- Utbyter routingtabeller mellan routrar.
  - Skickar och tar emot meddelanden på sina ‘interface’,
  - möjliggör för routrar att lära om andra nät,
  - informerar om förändringar i topologin.
- Beräknar väg utifrån den routing-information som är mottagen.
- Adaptive routing protocols.



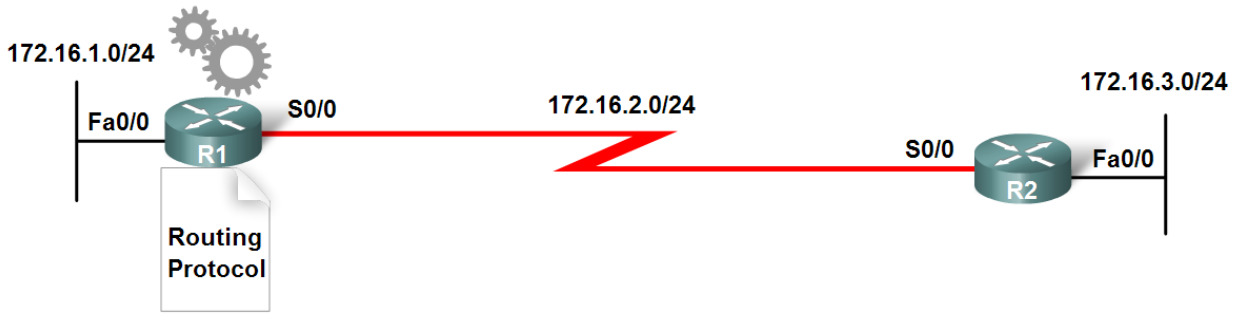

Figur 1: Dynamiskrouting[2]

## Sammanfattning algoritm

En ordnad uppsättning av tydligt definierade steg för att åstadkomma en viss uppgift.

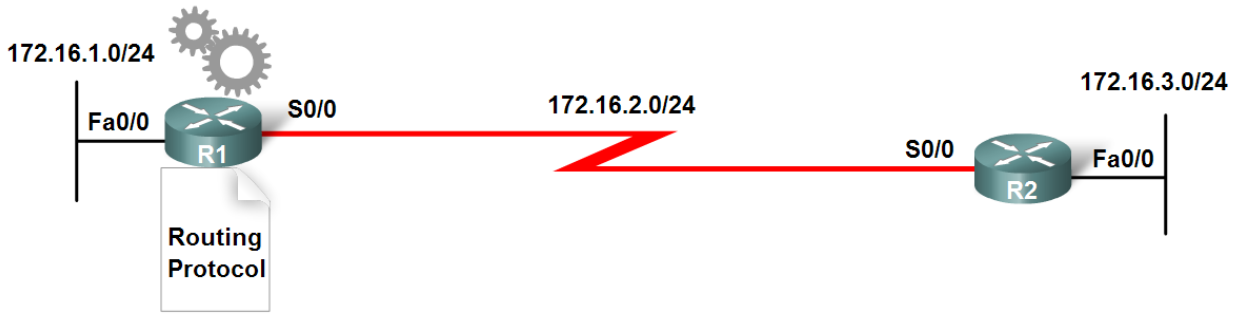
- Syftet med en routingalgoritm:
  - Mekanism för att skicka och ta emot routinginformation.
  - Mekanism för att beräkna den bästa vägen, och installera 'routes' i en routing-tabell.
  - Mekanism för att upptäcka och agera på förändringar i topologin.

- Datastruktur: Lagra routinginformation
- Meddelande: Utbyter routinginformation
- Algoritm: Bearbetar routinginformation, best-path calculation



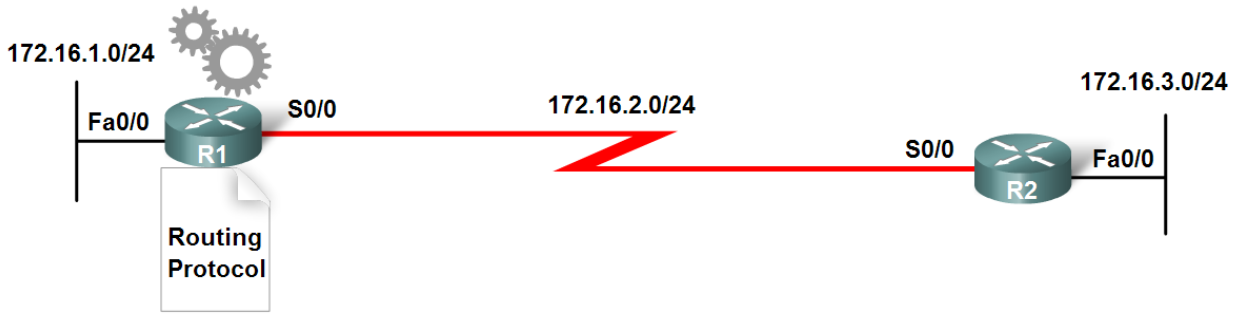

Figur 2: Dynamiska routinguppdateringar[2]

- Datastruktur: Lagra routinginformation
- Meddelande: Utbyter routinginformation
- Algoritm: Bearbetar routinginformation, best-path calculation

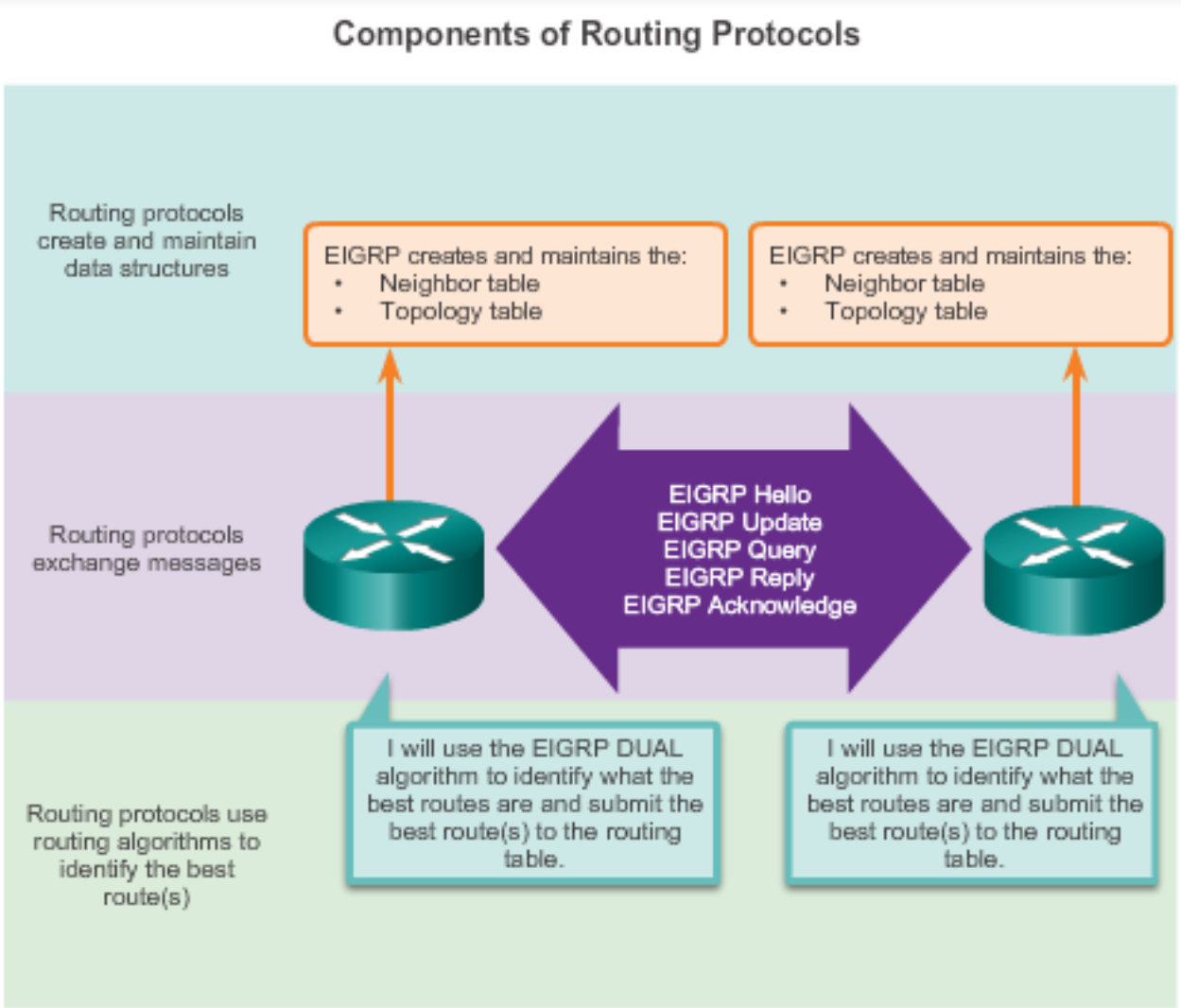



Figur 2: Dynamiska routinguppdateringar[2]

- Datastruktur: Lagra routinginformation
- Meddelande: Utbyter routinginformation
- Algoritm: Bearbetar routinginformation, best-path calculation

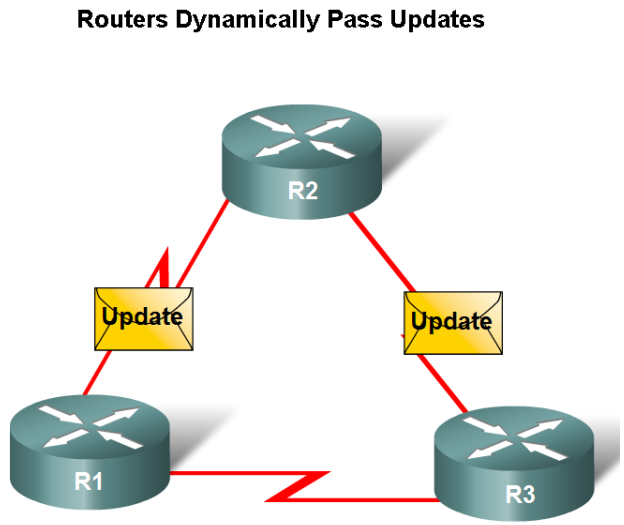



Figur 2: Dynamiska routinguppdateringar[2]



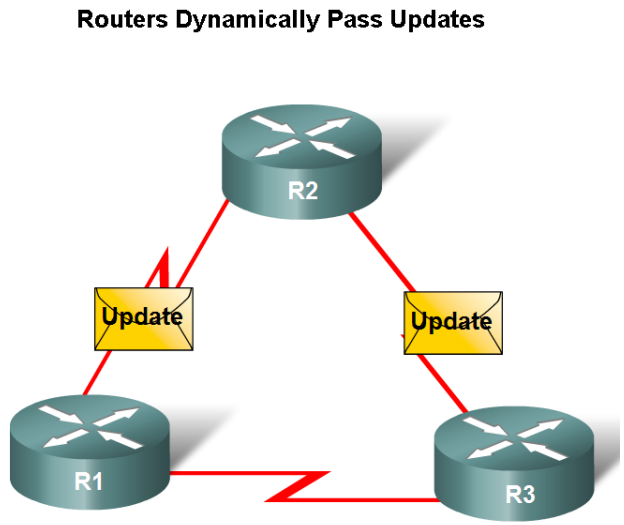

Figur 3: Dynamiskrouting[1]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),



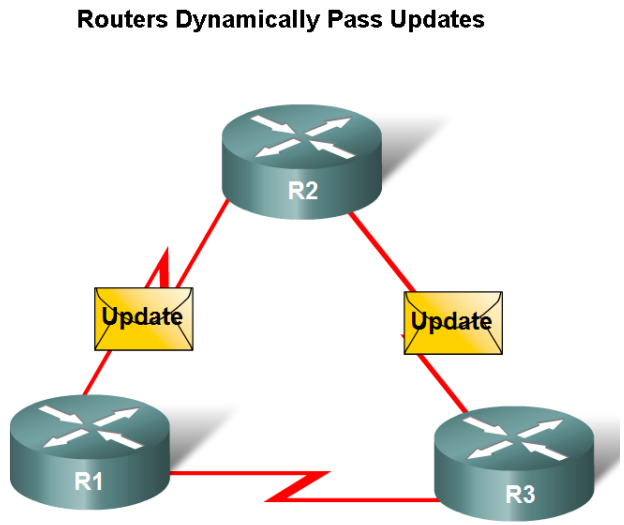

Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),



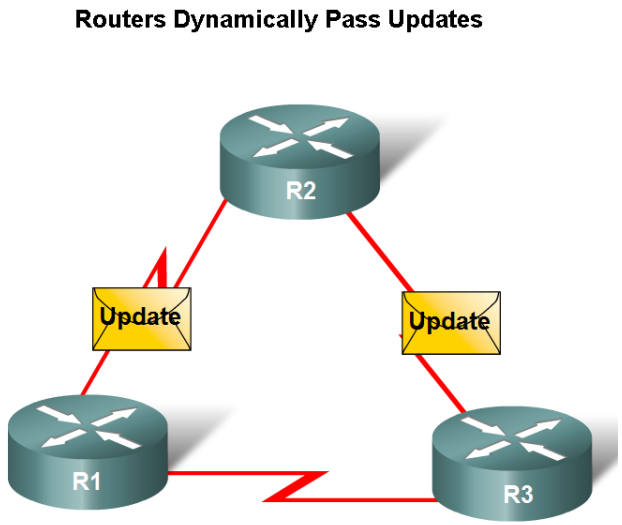

Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),



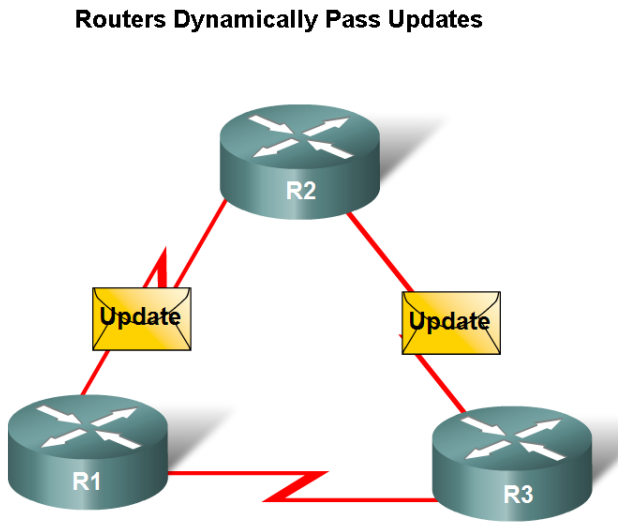

Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),



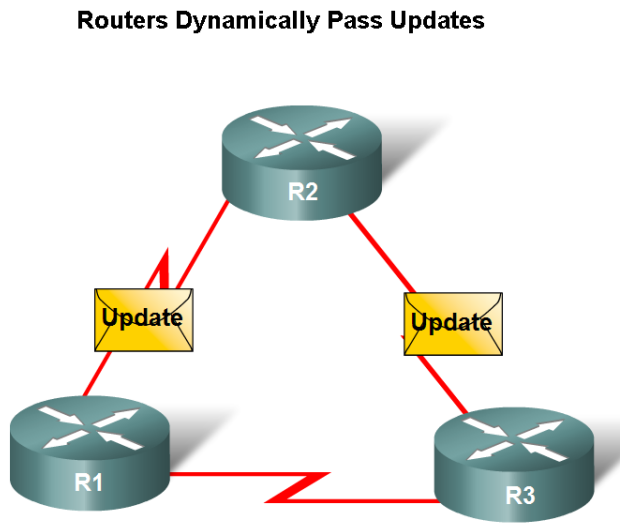

Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),



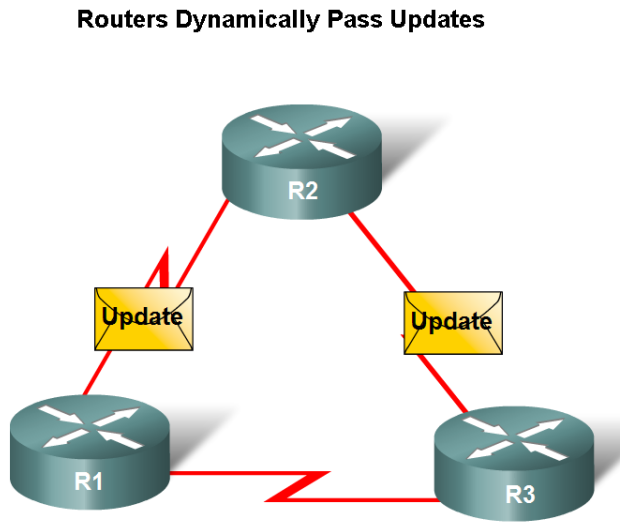

Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),




Figur 4: Dynamisk routinguppdatering [2]

- Fördelar
  - Liten administrativ 'overhead',
  - anpassar sig automatiskt till topologi-förändringar,
  - oberoende av nätverksstorlek,
  - beräknar ut bästa väg enligt parametrar.
- Nackdelar
  - Hög belastning (CPU, Bandbredd),




Figur 4: Dynamisk routinguppdatering [2]

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

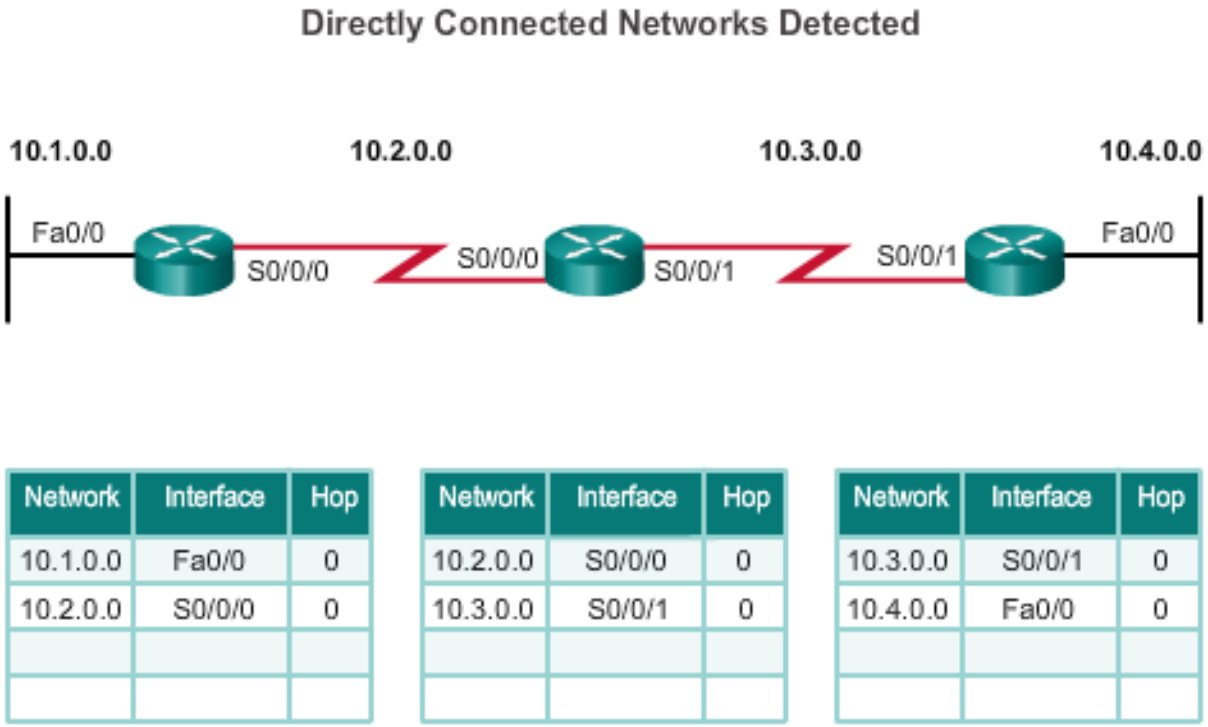
- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

## Vilka egenskaper vill vi se i en routingalgorithm?

- Korrekt,
  - Säkerställa korrekta resultat.
- Stabilitet,
  - Snabbt nå ett konvergerat tillstånd. *convergence*
  - Ska inte förändras om ej nödvändigt.
- Rättvist
  - Fördela tillgänglig bandbredd rättvist.
  - Inte att blanda ihop med lika.
  - Paretooptimalitet
- Simpelt,
  - Hög komplexitet.
  - Buggar.
  - Svårt att felsöka.
- Robust,
  - Ska kunna motstå konstanta förändringar i nätverket.
- Effektivt.
  - Minska antalet hopp,
  - skicka data över lågt belastade länkar,
  - maximera genomströmning.

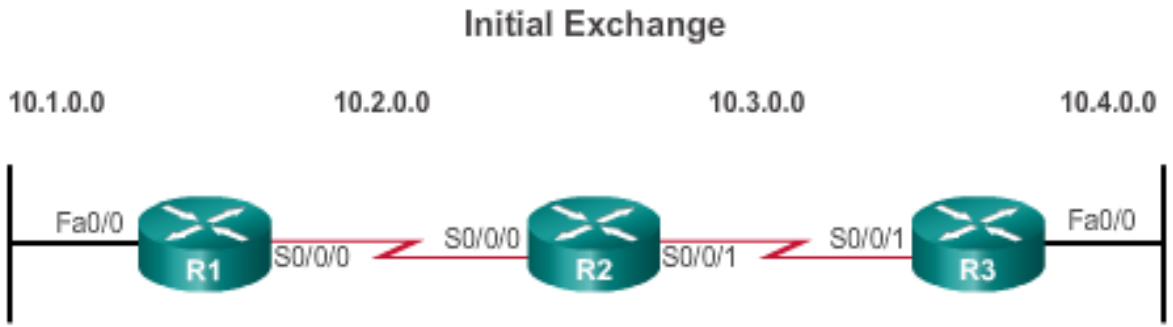



| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| | | |
| | | |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| | | |
| | | |

| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/1 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| | | |
| | | |

Figur 5: Kallstart [1]

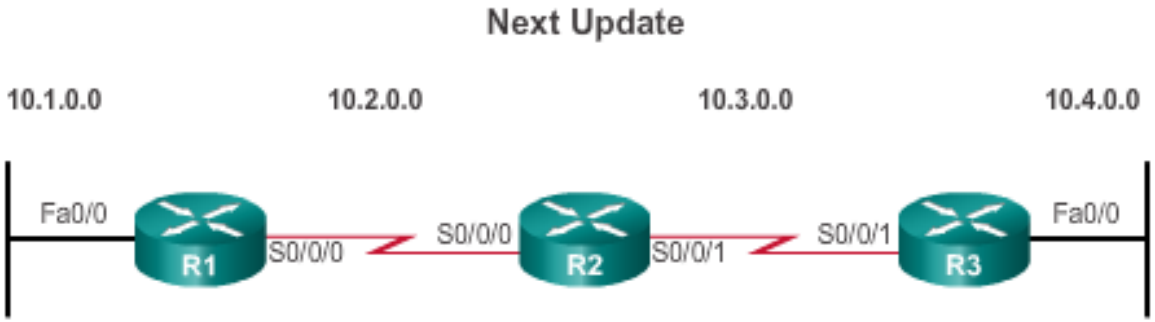



| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/0 | 1 |
| | | |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| 10.1.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/1 | 1 |

| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/0 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/1 | 1 |
| | | |

Figur 6: Första informationsutbytet [1]




| Network | Interface | Hop |
|---|---|---|
| 10.1.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/0 | 2 |

| Network | Interface | Hop |
|---|---|---|
| 10.2.0.0 | S0/0/0 | 0 |
| 10.3.0.0 | S0/0/1 | 0 |
| 10.1.0.0 | S0/0/0 | 1 |
| 10.4.0.0 | S0/0/1 | 1 |

| Network | Interface | Hop |
|---|---|---|
| 10.3.0.0 | S0/0/1 | 0 |
| 10.4.0.0 | Fa0/0 | 0 |
| 10.2.0.0 | S0/0/1 | 1 |
| 10.1.0.0 | S0/0/1 | 2 |

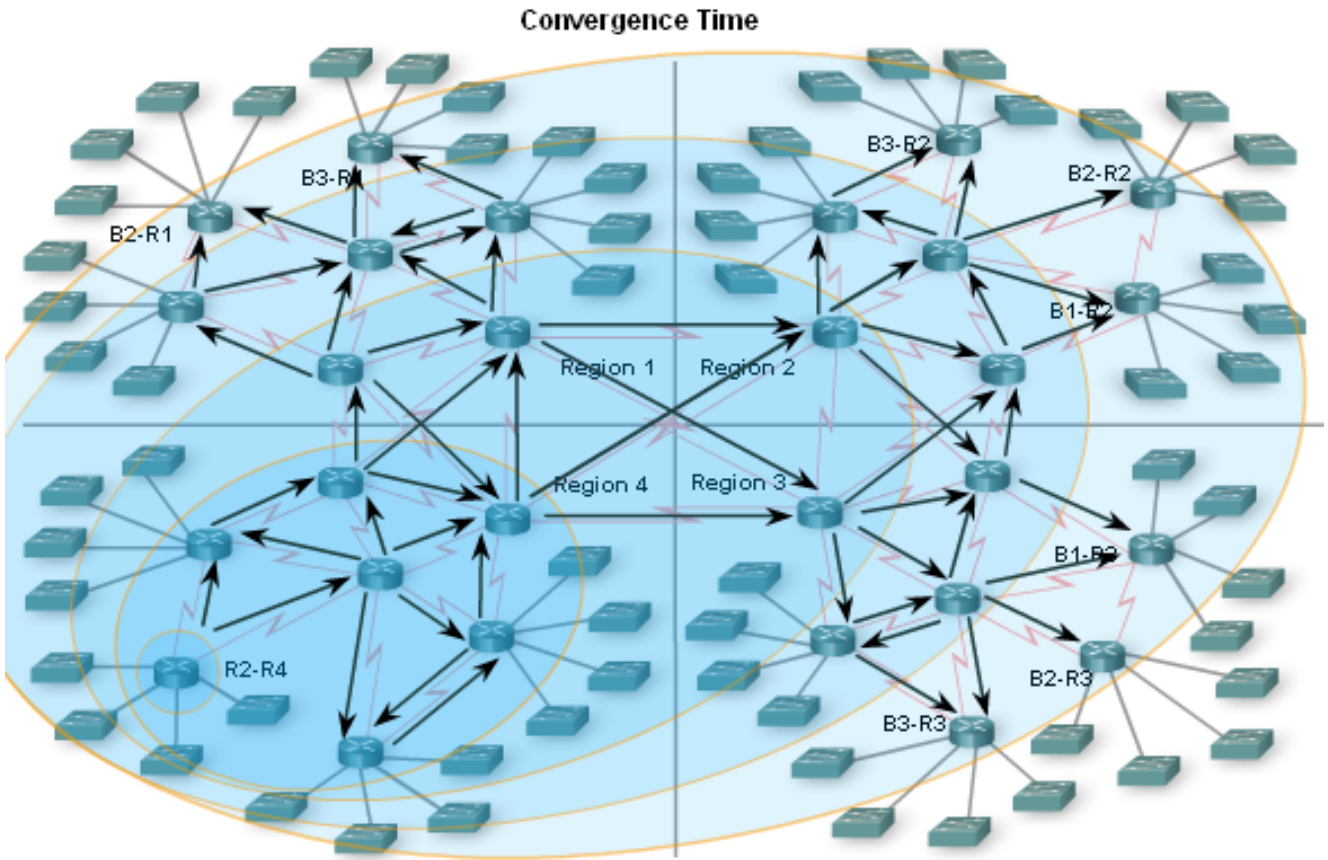
Figur 7: Andra informationsutbytet [1]

- Konvergeringstid
- Tid beroende på flertal faktorer:
  - Utbredningstid.
  - Beräkningstid.

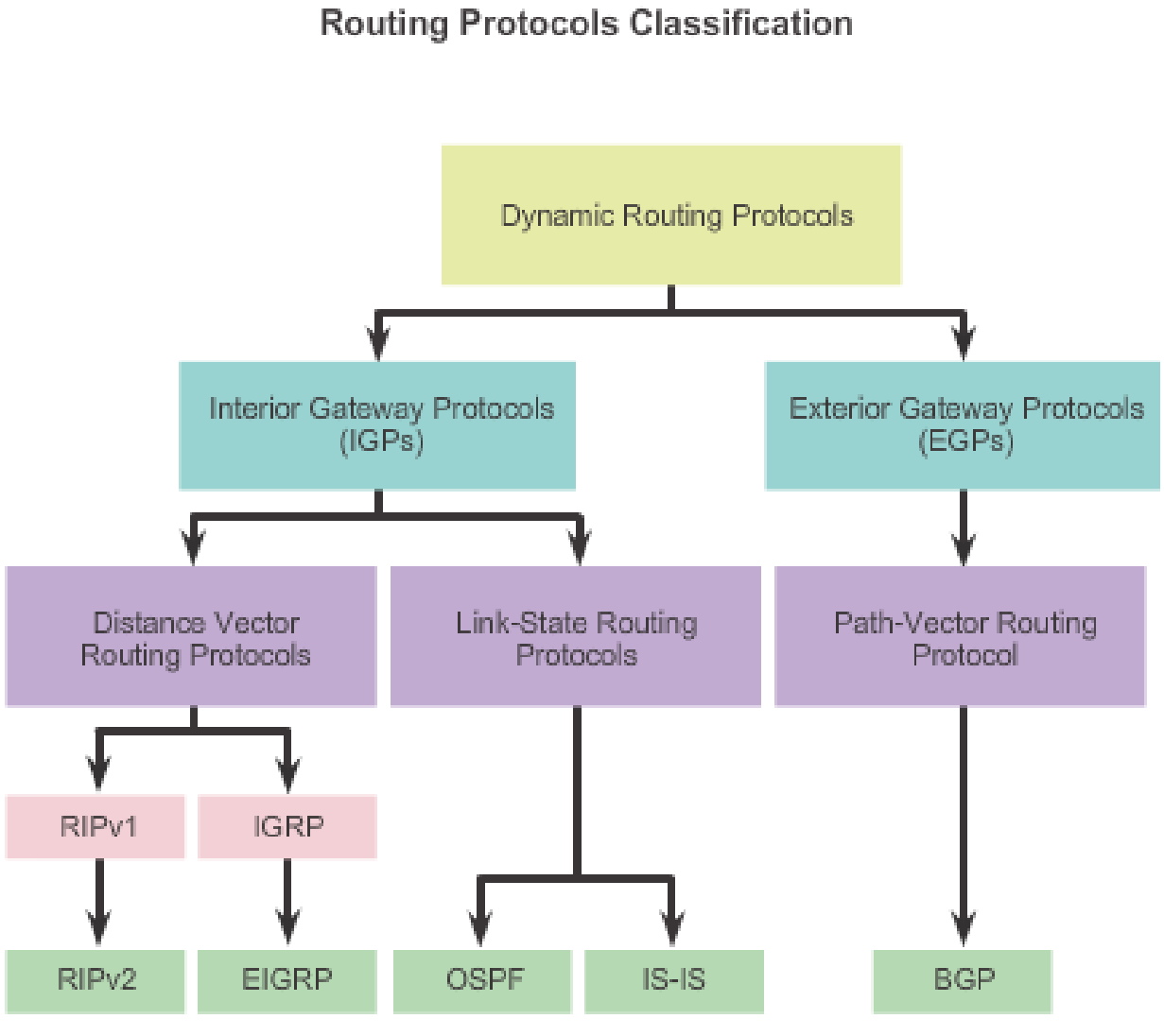
- Konvergeringstid
- Tid beroende på flertal faktorer:
  - Utbredningstid.
  - Beräkningstid.

- Konvergeringstid
- Tid beroende på flertal faktorer:
  - Utbredningstid.
  - Beräkningstid.

- Konvergeringstid
- Tid beroende på flertal faktorer:
  - Utbredningstid.
  - Beräkningstid.




Figur 8: Konvergeringstid [2]




Figur 9: Klassifiera routingprotokoll [1]

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

## Jämföra routingprotokoll

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

## Jämföra routingprotokoll

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

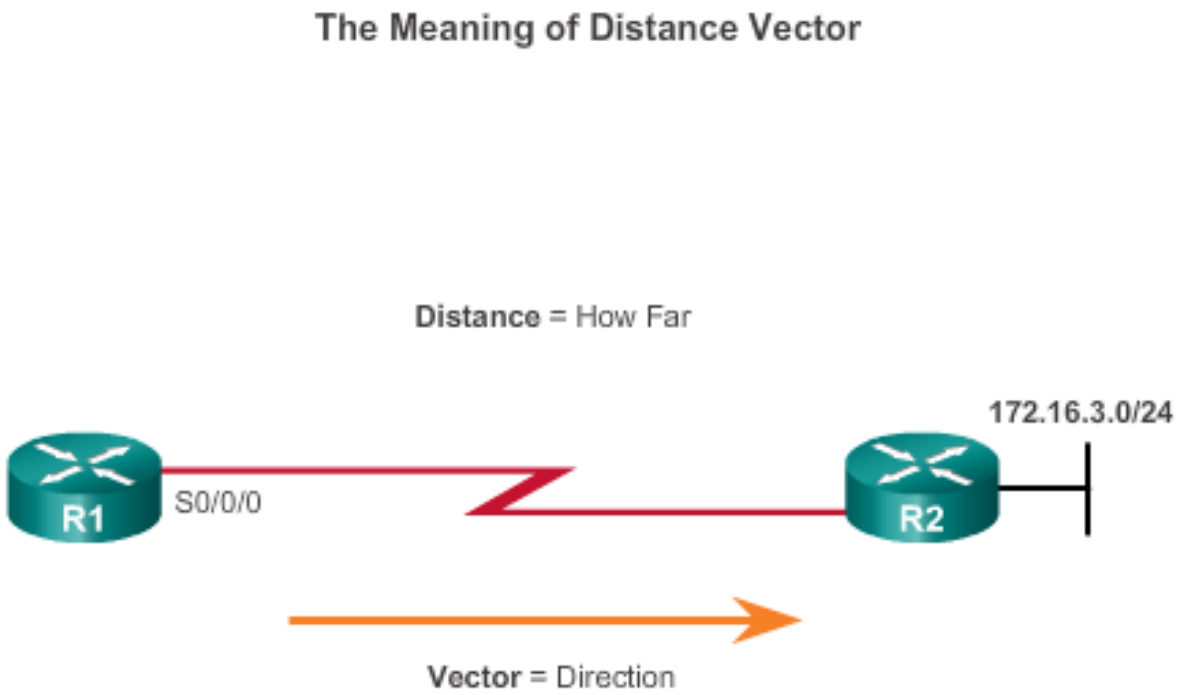
- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

- Konvergeringstid
  - Tid innan samtliga routers i ett nätverk har komplett information.
- Skalbarhet
  - Hur stort nätverk kan protokollet stöda.
- Klasser eller inte klasser?
- Resursanvändning
  - CPU
  - Länkbandbredden.
  - Minnesanvändning
- Implementering och underhåll.
  - Kunskap som krävs för att implementera, underhålla och felsöka.

# Innehållsförteckning




Figur 10: Distance Vector [1]

## Distance

- Anger distansen till ett mål.
- I vilken enhet är ej angivet.

## Vector

- Anger en riktning till ett mål.

## Distance

- Anger distansen till ett mål.
- I vilken enhet är ej angivet.

## Vector

- Anger en riktning till ett mål.

## Navigation

Ta sig till ett mål enbart med hjälp utav vägskyltar.


Figur 11: Vägskylt [3]

## Karaktärsdrag

- Skickar uppdateringar (periodiska) med de routrar som är direkt anslutna.
- Saknar information om hur topologin ser ut.

## Karaktärsdrag

- Skickar uppdateringar (periodiska) med de routrar som är direkt anslutna.
- Saknar information om hur topologin ser ut.

## Uppdateringar

- RIPv1 skickar uppdateringar med destination 255.255.255.255
- RIPv2 och EIGRP använder Multicast (224.0.0.9, 224.0.0.10).
- EIGRP skickar enbart uppdateringar vid behov.

## Uppdateringar

- RIPv1 skickar uppdateringar med destination 255.255.255.255
- RIPv2 och EIGRP använder Multicast (224.0.0.9, 224.0.0.10).
- EIGRP skickar enbart uppdateringar vid behov.

## Uppdateringar

- RIPv1 skickar uppdateringar med destination 255.255.255.255
- RIPv2 och EIGRP använder Multicast (224.0.0.9, 224.0.0.10).
- EIGRP skickar enbart uppdateringar vid behov.

# Innehållsförteckning

- Ultimate Route
- Level 1 Route
- Level 1 Parent Route
- Level 2 Child Route

Routing Table of R1

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0
L        172.16.1.1/32 is directly connected, GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
      Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C        209.165.200.224/30 is directly connected, Serial0/0/0
L        209.165.200.225/32 is directly connected, Serial0/0/0
R        209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
C        209.165.200.232/30 is directly connected, Serial0/0/1
L        209.165.200.233/32 is directly connected, Serial0/0/1
R1#
```

Figur 12: Routingtabell [1]

- Ultimate Route
- Level 1 Route
- Level 1 Parent Route
- Level 2 Child Route

Routing Table of R1

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0
L        172.16.1.1/32 is directly connected, GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
      Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C        209.165.200.224/30 is directly connected, Serial0/0/0
L        209.165.200.225/32 is directly connected, Serial0/0/0
R        209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
C        209.165.200.232/30 is directly connected, Serial0/0/1
L        209.165.200.233/32 is directly connected, Serial0/0/1
R1#
```

Figur 12: Routingtabell [1]

- Ultimate Route
- Level 1 Route
- Level 1 Parent Route
- Level 2 Child Route

Routing Table of R1

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0
L        172.16.1.1/32 is directly connected, GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
      Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C        209.165.200.224/30 is directly connected, Serial0/0/0
L        209.165.200.225/32 is directly connected, Serial0/0/0
R        209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
C        209.165.200.232/30 is directly connected, Serial0/0/1
L        209.165.200.233/32 is directly connected, Serial0/0/1
R1#
```

Figur 12: Routingtabell [1]

- Ultimate Route
- Level 1 Route
- Level 1 Parent Route
- Level 2 Child Route

Routing Table of R1

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0
L        172.16.1.1/32 is directly connected, GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
      Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C        209.165.200.224/30 is directly connected, Serial0/0/0
L        209.165.200.225/32 is directly connected, Serial0/0/0
R        209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
C        209.165.200.232/30 is directly connected, Serial0/0/1
L        209.165.200.233/32 is directly connected, Serial0/0/1
R1#
```

Figur 12: Routingtabell [1]

- Anger Next-Hop, Interface

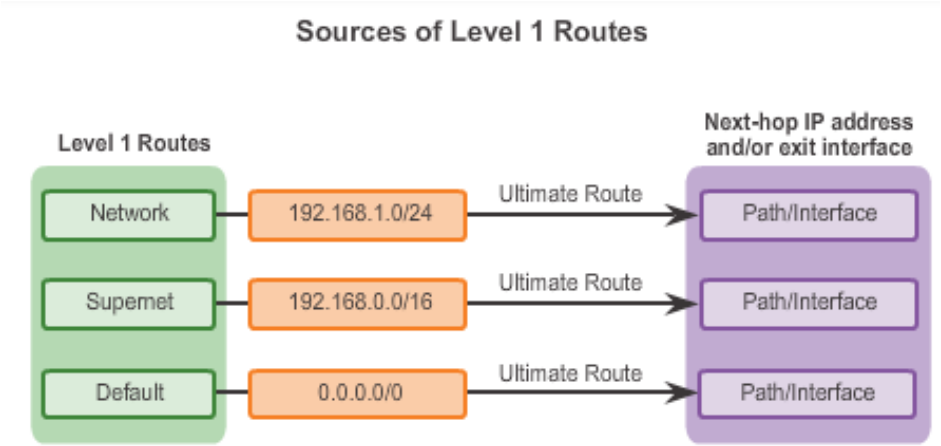
Ultimate Routes of R1

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0
L        172.16.1.1/32 is directly connected, GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12,
          Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12,
         Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
      Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C        209.165.200.224/30 is directly connected, Serial0/0/0
L        209.165.200.225/32 is directly connected, Serial0/0/0
R        209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
         Serial0/0/0
C        209.165.200.232/30 is directly connected, Serial0/0/1
L        209.165.200.233/32 is directly connected, Serial0/0/1
R1#
```

Figur 13: Ultimate Route [1]

Subnetmasken är mindre än eller lika med motsvarande klass-mask.




Figur 14: Level 1 Route [1]

Ett L1 nätverk som är uppdelad i subnät (child).

**Level 1 Parent Routes of R1**

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network
0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3
masks
C        172.16.1.0/24 is directly connected,
GigabitEthernet0/0
L        172.16.1.1/32 is directly connected,
GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226,
00:00:12, Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226,
00:00:12, Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226,
00:00:12, Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2
masks
C        209.165.200.224/30 is directly connected,
Serial0/0/0
```

Figur 15: Level 1 Parent Route [1]

Ett subnät inom ett Level 1 nätverk.

**Example of Level 2 Child Routes**

```
R1#show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network
0.0.0.0

S*    0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
                is directly connected, Serial0/0/1
      172.16.0.0/16 is variably subnetted, 5 subnets, 3
masks
C        172.16.1.0/24 is directly connected,
GigabitEthernet0/0
L        172.16.1.1/32 is directly connected,
GigabitEthernet0/0
R        172.16.2.0/24 [120/1] via 209.165.200.226,
00:00:12, Serial0/0/0
R        172.16.3.0/24 [120/2] via 209.165.200.226,
00:00:12, Serial0/0/0
R        172.16.4.0/28 [120/2] via 209.165.200.226,
00:00:12, Serial0/0/0
R     192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03,
Serial0/0/0
      209.165.200.0/24 is variably subnetted, 5 subnets, 2
masks
C        209.165.200.224/30 is directly connected,
Serial0/0/0
```

Figur 16: Level 2 Child Route [1]

- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.

- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.

- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.

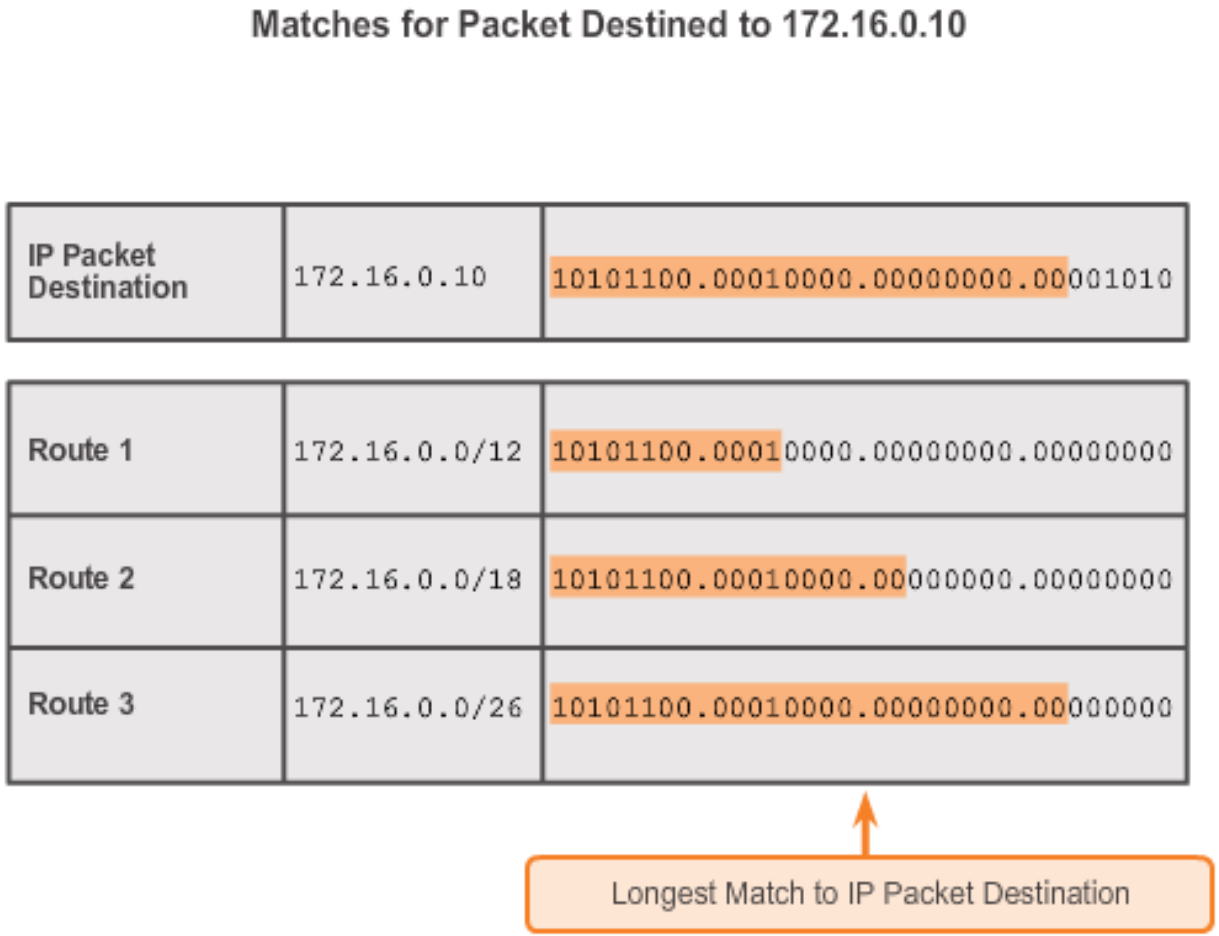
- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.

## Uppslagningsprocessen

- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.

## Uppslagningsprocessen

- Om den bästa matchningen är en L1 Ultimate Route, används denna.
- Om den bästa matchningen är en L1 Parent Route:
  - Undersök varje Child Route för denna Parent Route.
  - Hittas en matchning används denna.
- Söker nästa L1 supernet, inklusive default route.
- Hittas ingen träff, kastas paketet.




Figur 17: Bästa vägen = Längsta matchning [1]

## Referenser



Scott Empson och Cheryl Schmidt. *Routing and Switching Essentials – Companion Guide*. Cisco Press, 2014. ISBN: 978-1-58713-320-6.


Rick Graziani och Allan Johnson. *Routing protocols and concepts : CCNA exploration companion guide*. Indianapolis, Ind.: Cisco, 2008. ISBN: 9781587132063 (hbk.)


Holger.Ellgaard. *Vägskylt Södertälje Stockholm 2011*. Creative Commons Attribution-ShareAlike 3.0. 2011. URL: http://commons.wikimedia.org/wiki/File:V%C3%A4gskylt_S%C3%B6dert%C3%A4lje_Stockholm_2011.jpg.