

---

# Rechnernetze I

SoSe 2020

Roland Wismüller  
Universität Siegen  
roland.wismueller@uni-siegen.de  
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 12. März 2020

---

# Rechnernetze I

SoSe 2020

## 4 LAN Switching



## Inhalt

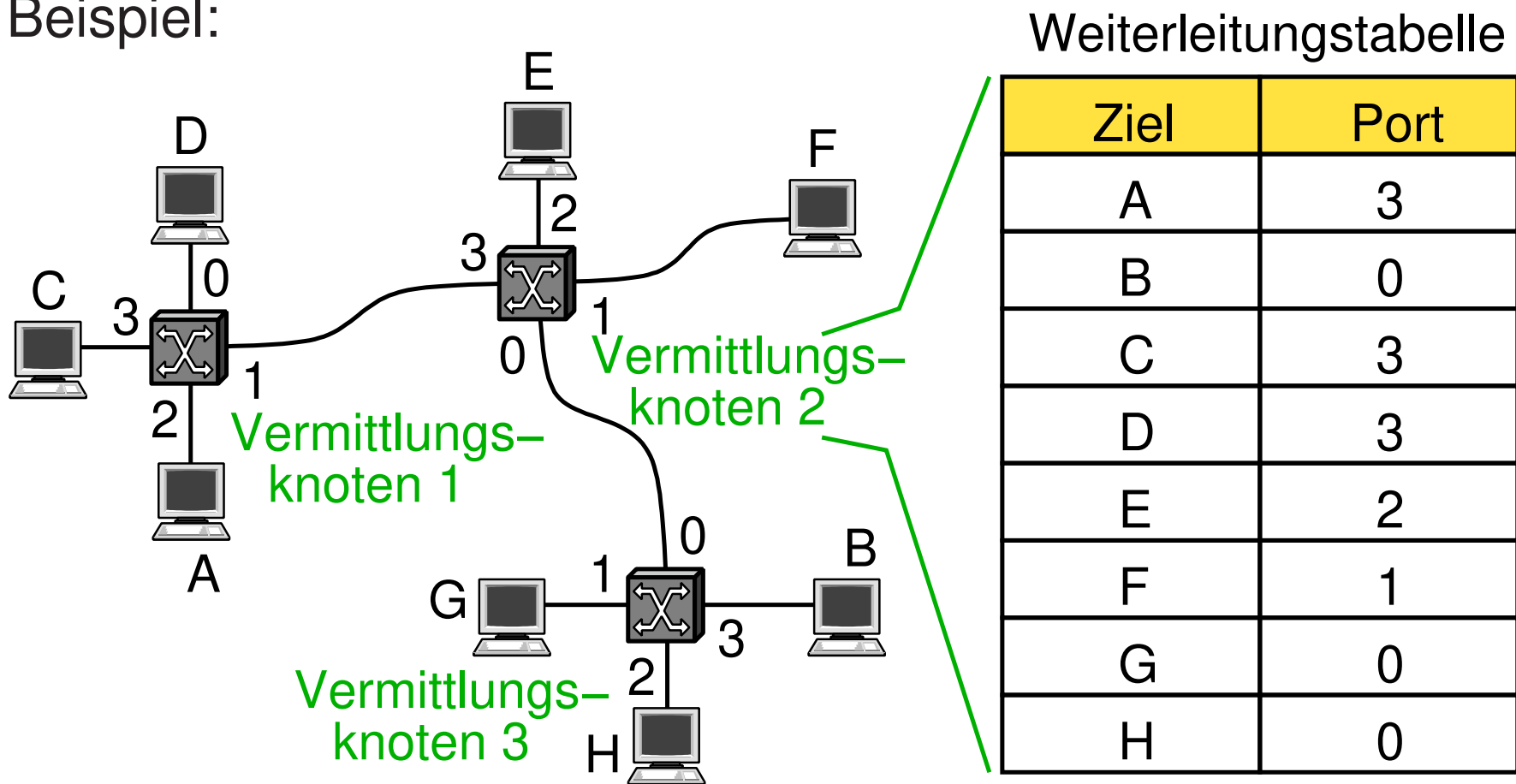
- ➔ Weiterleitungstechniken
- ➔ Switching: Einführung
- ➔ Lernende Switches
- ➔ *Spanning-Tree-Algorithmus*
- ➔ Virtuelle LANs
  
- ➔ Peterson, Kap. 3.1, 3.2, 3.4
- ➔ CCNA, Kap. 5.2



## Weiterleitung von Datagrammen (verbindungslos)

- ➔ Jeder Vermittlungsknoten besitzt eine Weiterleitungstabelle
  - ➔ bildet Zieladresse auf Ausgangsport ab

➔ Beispiel:





### Weiterleitung von Datagrammen: Eigenschaften

- ➔ Knoten können jederzeit ein Paket an andere Knoten senden; Pakete können sofort weitergeleitet werden
  - ➔ kein Verbindungsaufbau
- ➔ Ein Knoten kann nicht feststellen, ob das Netz das Paket zustellen kann
- ➔ Pakete werden unabhängig voneinander weitergeleitet
- ➔ Ausfall einer Verbindung bzw. eines Vermittlungsknotens kann prinzipiell toleriert werden
  - ➔ Anpassung der Weiterleitungstabellen
- ➔ Eingesetzt z.B. im Internet (IP)
- ➔ (vgl. Kap. 1.3: Paketvermittlung)

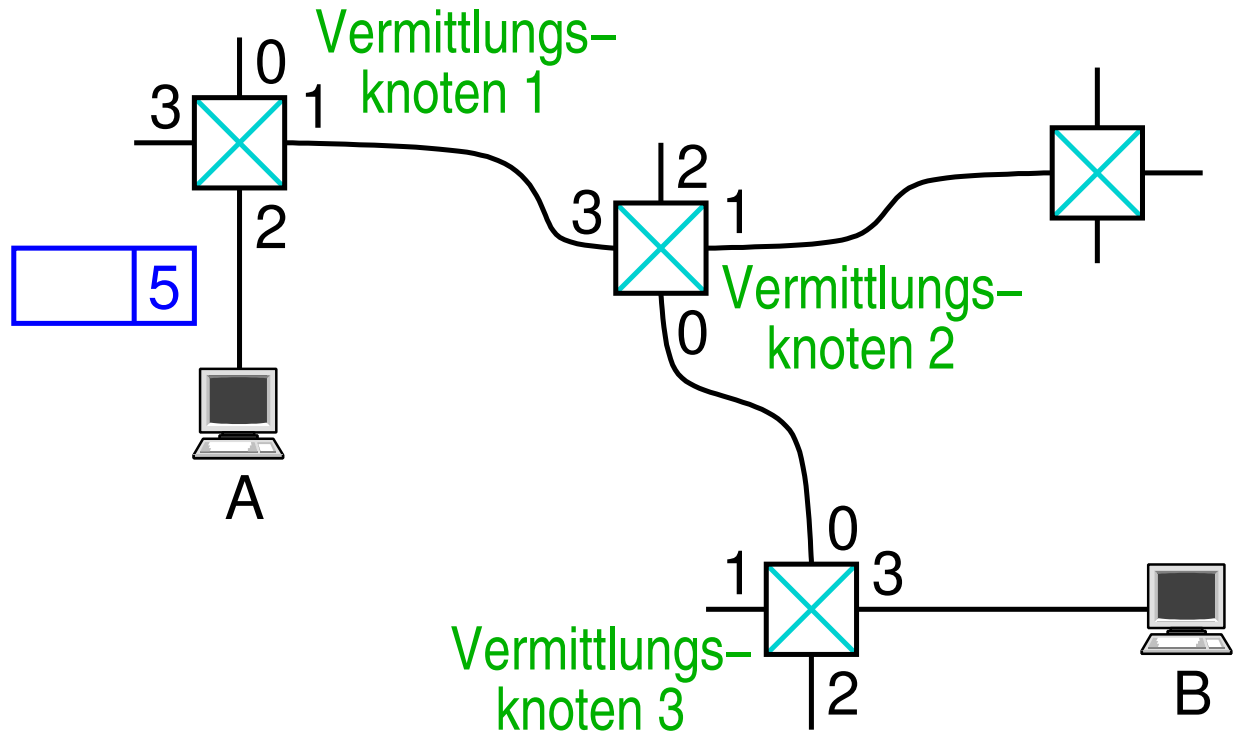
### Virtuelle Leitungsvermittlung (verbindungsorientiert)

- ➔ Kommunikation in zwei Phasen:
  - ➔ Aufbau einer virtuellen Verbindung (**Virtual Circuit, VC**) vom Quell- zum Zielrechner
    - ➔ statisch (*Permanent VC*)
    - ➔ dynamisch (*Switched VC*)
  - ➔ Versenden der Pakete über VC:
    - ➔ Pakete enthalten Bezeichner des VC
    - ➔ alle Pakete nehmen denselben Weg
- ➔ VC-Bezeichner (VCI, *Virtual Circuit Identifier*) nur auf den einzelnen Leitungen eindeutig
  - ➔ Weiterleitungstabelle im Vermittlungsknoten bildet Eingangs-Port und -VCI auf Ausgangs-Port und -VCI ab



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



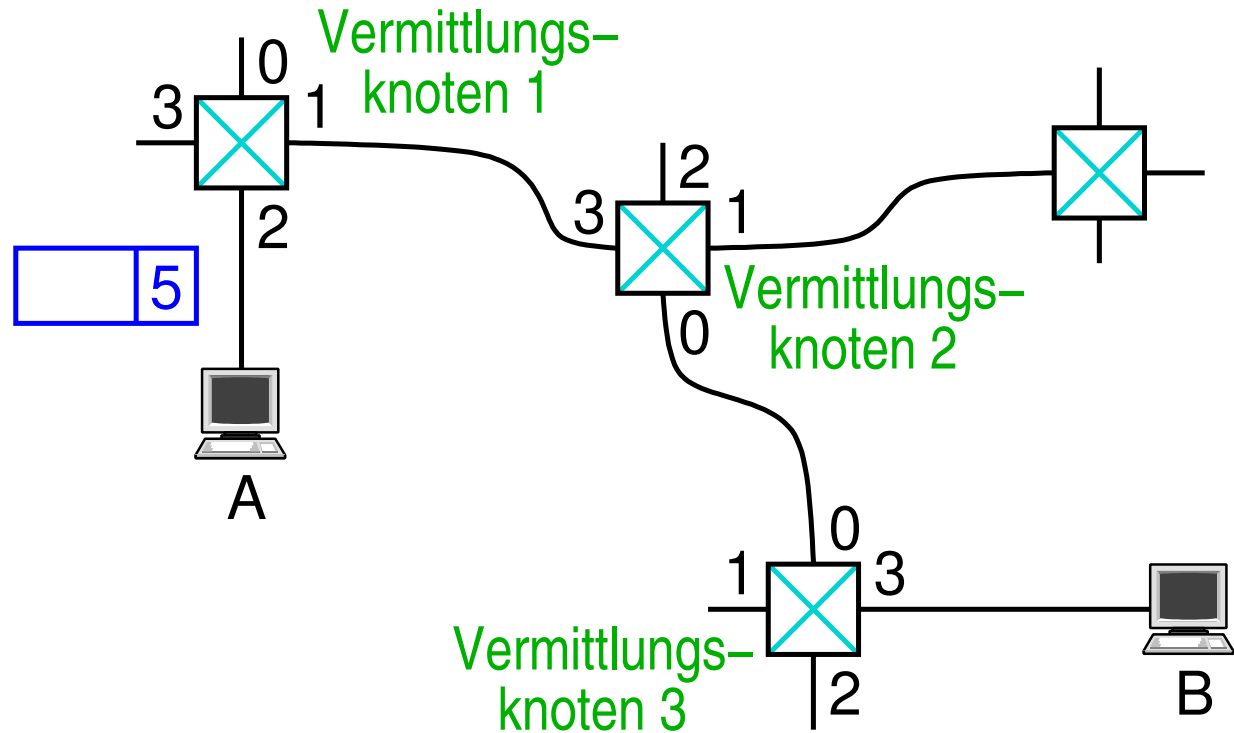
	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

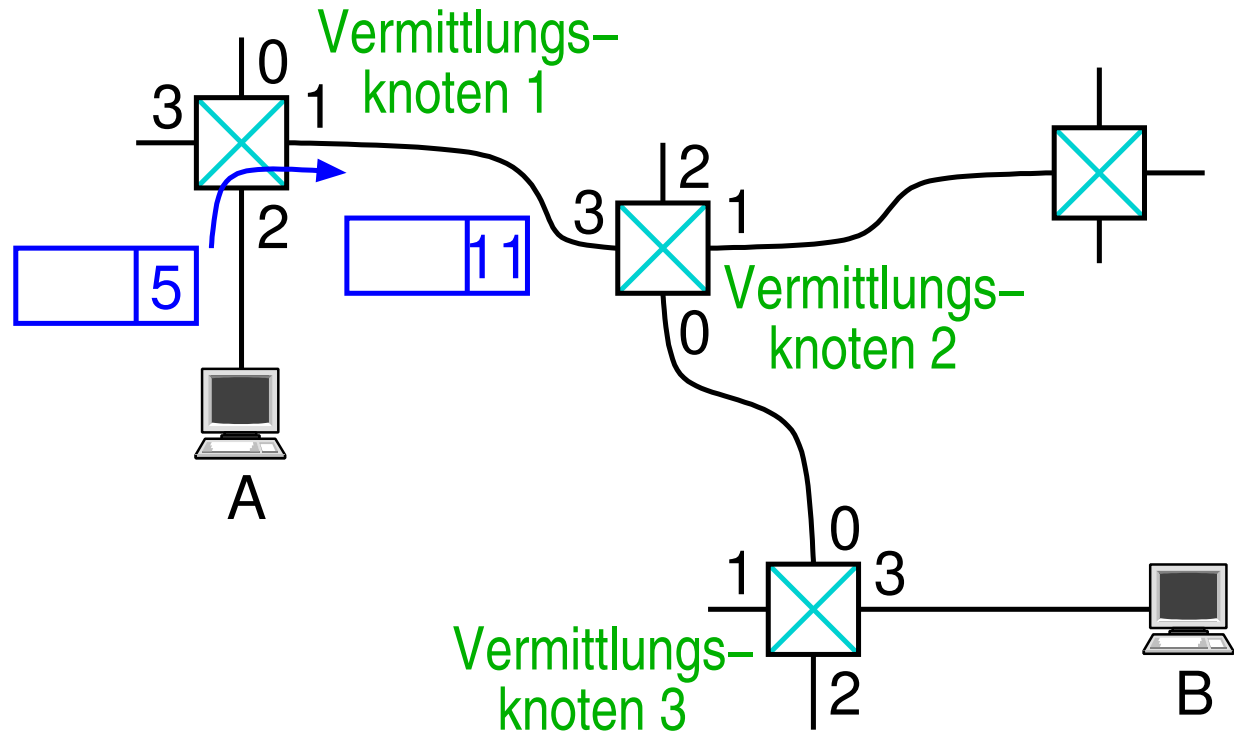
➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)





## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



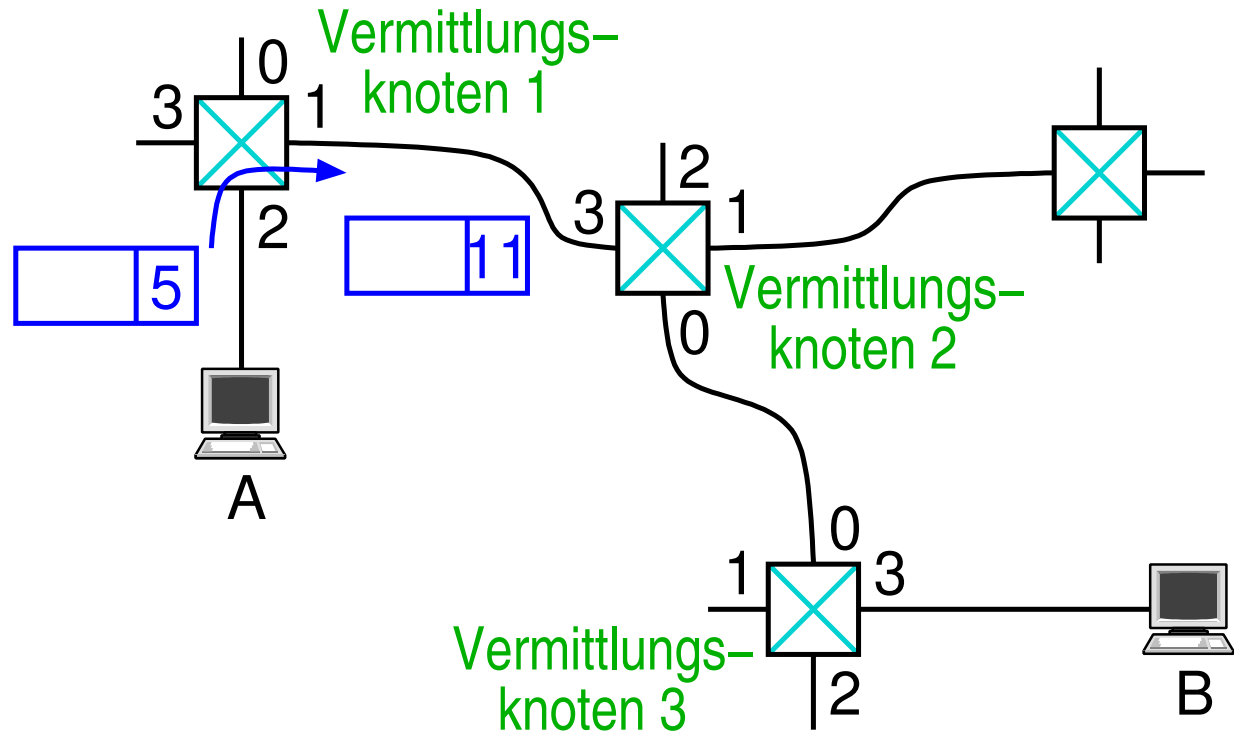
	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



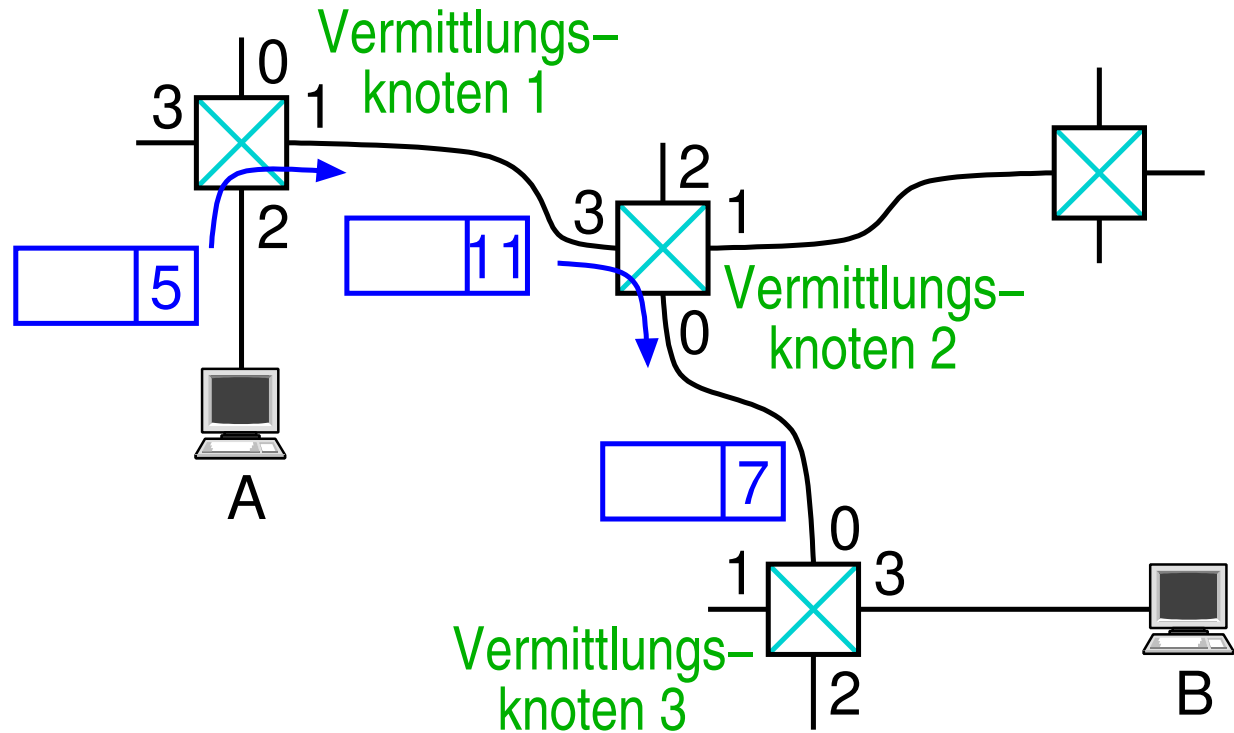
	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



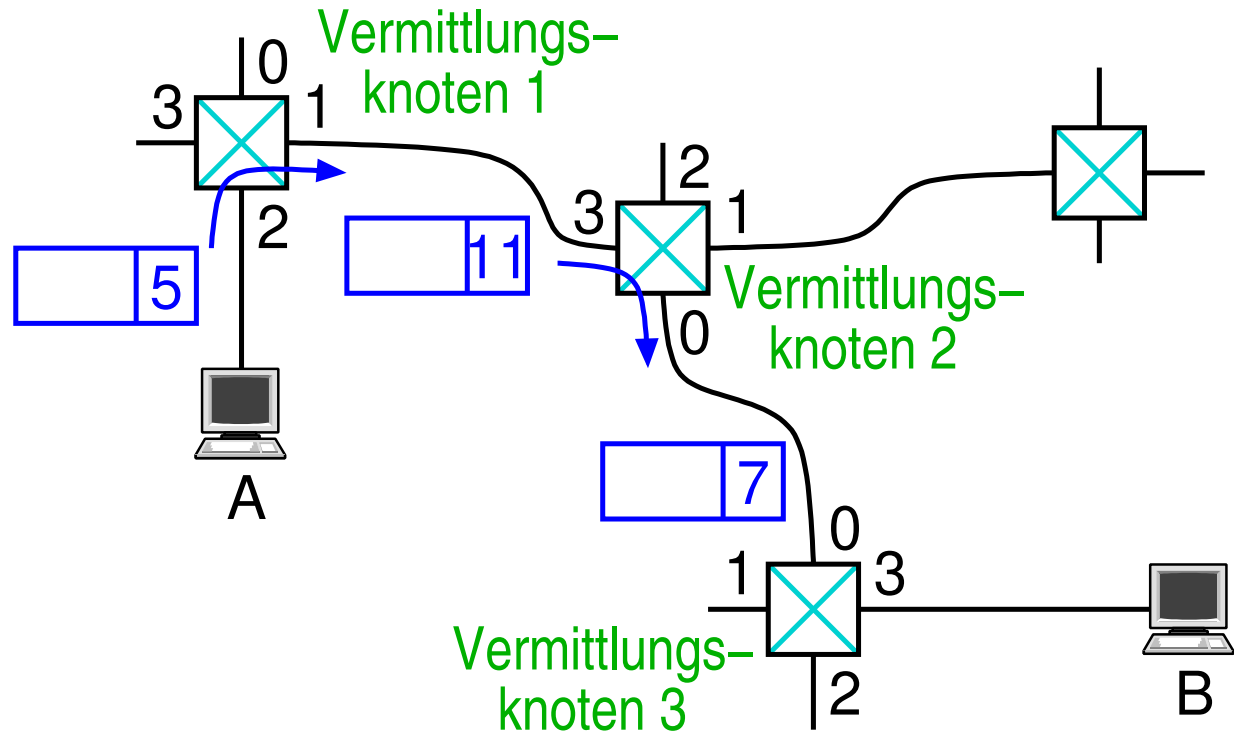
	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



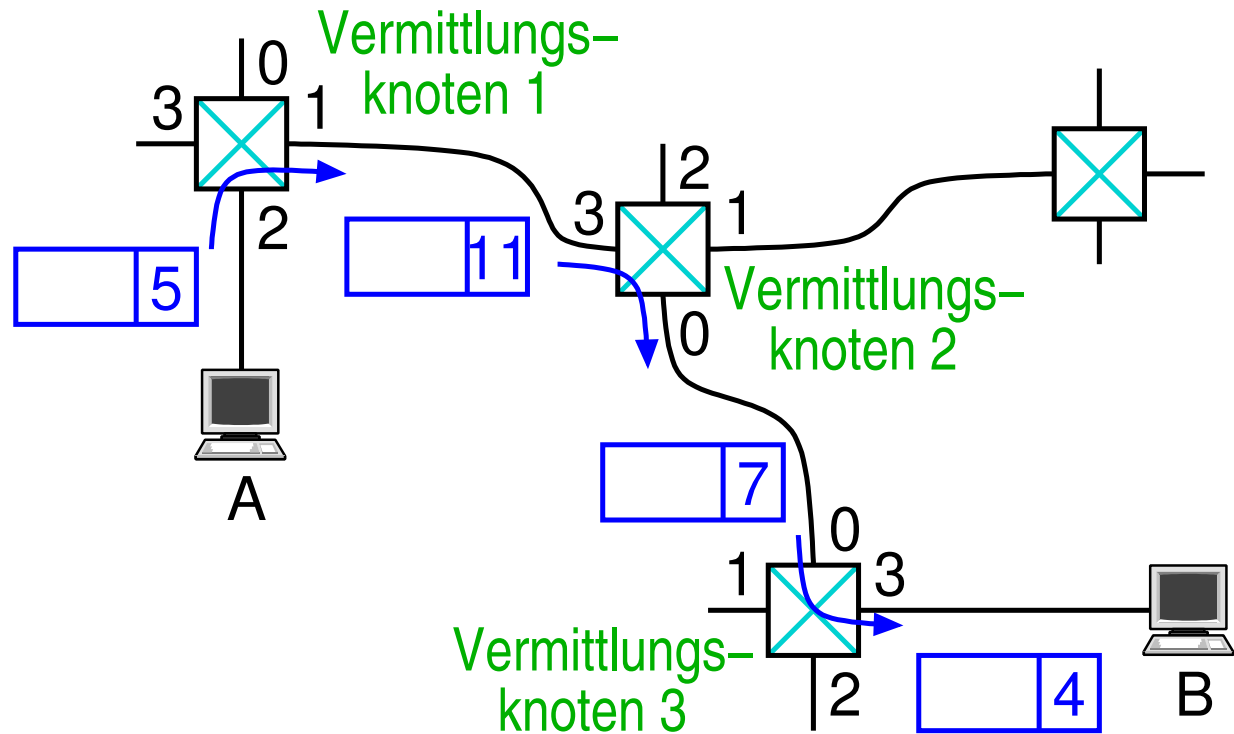
	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)



## Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

➔ Beispiel:  
A sendet an B



	Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

➔ Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)

## Vergleich der Weiterleitungstechniken

	<b>Datagramm-Verm.</b>	<b>Virtuelle Leitungsv.</b>
Verbindungsaufbau	nicht nötig	erforderlich
Adressierung	Pakete enthalten volle Sender- und Empfänger-Adresse	Pakete enthalten nur kurze VC-Bezeichner
Wegewahl	erfolgt unabhängig für jedes Paket	Weg wird bei Aufbau des VC festgelegt
Bei Ausfall eines Vermittlungsknotens	keine größeren Auswirkungen	alle VCs mit diesem Vermittlungsknoten sind unterbrochen
Dienstgütegarantien (QoS) und Überlastkontrolle	schwierig	Einfach, wenn vorab Ressourcen für VC reserviert werden



### Begriffe

#### ➔ **Switching / Forwarding (Weiterleitung):**

- ➔ Weiterleiten v. Frames (Paketen) zum richtigen Ausgangsport

#### ➔ **(LAN-)Switch / Bridge (Brücke):**

- ➔ Vermittler im LAN (auf Ebene der Sicherungsschicht)
- ➔ Bridge: Switch mit nur zwei Ports

#### ➔ **Routing:**

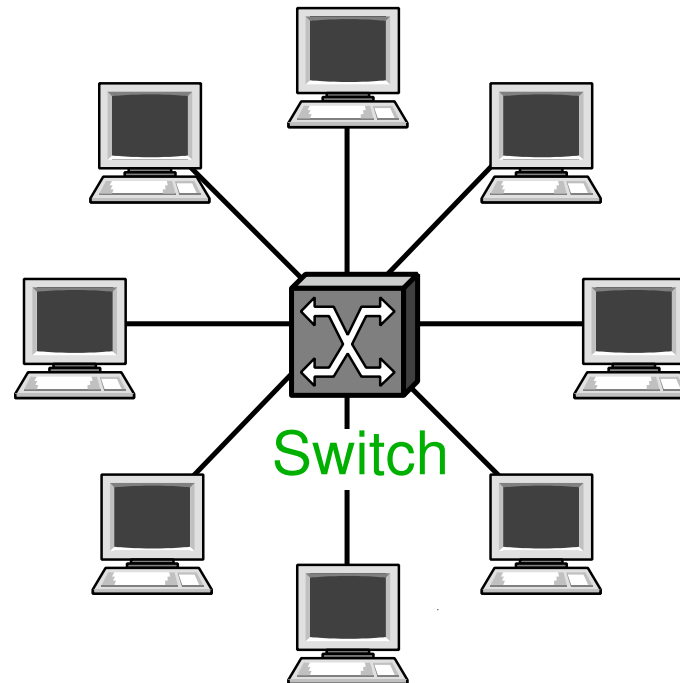
- ➔ (dynamischer) Aufbau von Tabellen zum Forwarding
- ➔ Ziel: Finden von (guten/optimalen) Wegen zu anderen Netzen

#### ➔ **Router:**

- ➔ Knoten, der mit den Protokollen der Vermittlungsschicht Pakete weiterleitet
- ➔ vereinigt Funktionalität von Routing und Forwarding



- ➔ Motivation: Ersetzen von Mehrfachzugriffsverbindungen im LAN durch Punkt-zu-Punkt-Verbindungen
- ➔ Vermittler (Switch):
  - ➔ mehrere Ein-/Ausgänge
  - ➔ leitet Frames aufgrund der Adressen im Header weiter
- ➔ Führt zu Sterntopologie:

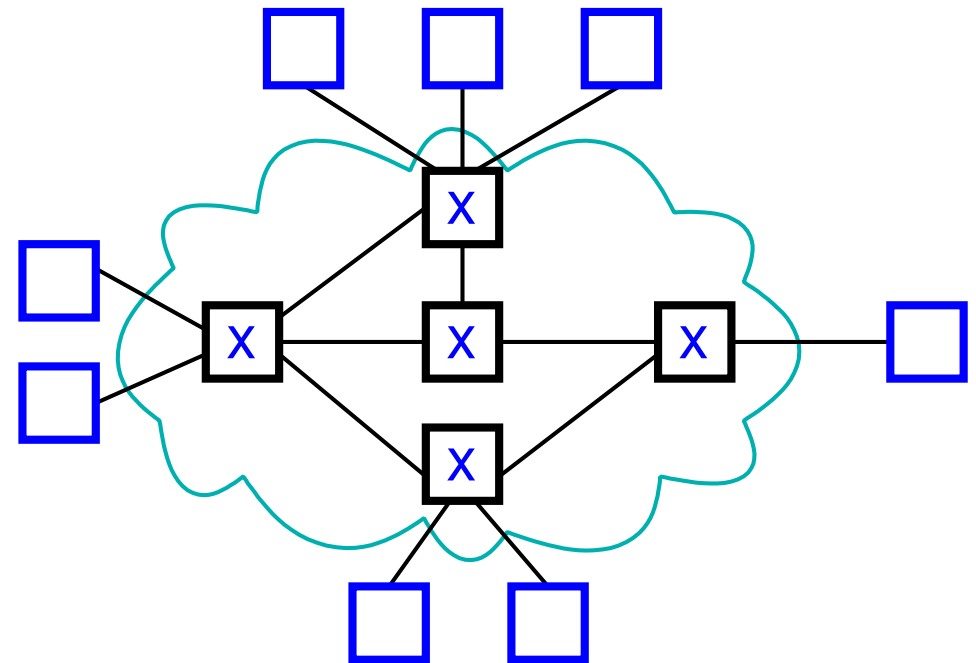




## 4.2 Switching: Einführung ...



- ➔ Vorteil gegenüber Bustopologie:
  - ➔ Kommunikation zwischen zwei Knoten wirkt sich nicht (notwendigerweise) auf andere Knoten aus
- ➔ Beschränkungen können durch Zusammenschalten mehrerer Switches überwunden werden:
  - ➔ begrenzte Anzahl von Ein-/Ausgängen pro Switch
  - ➔ Leitungslänge, geographische Ausdehnung



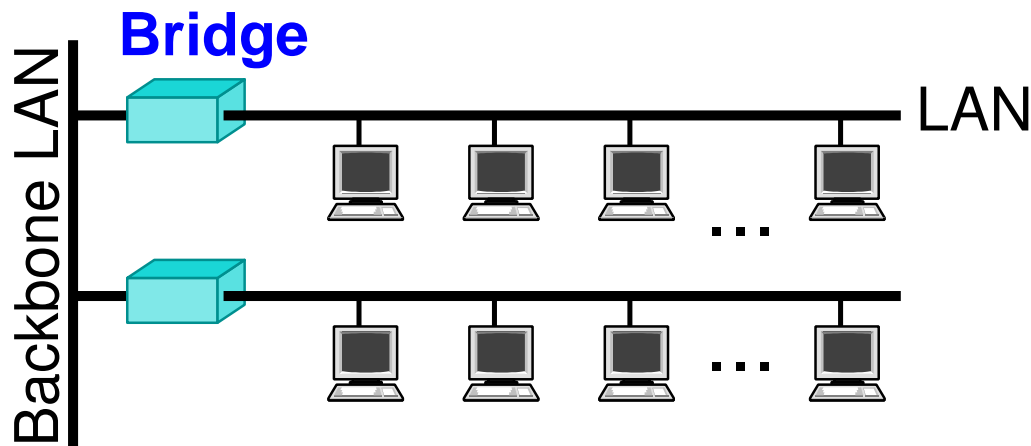


- ➔ Weitere Beschränkungen von Switches
  - ➔ Heterogenität
    - ➔ die verbundenen Netze müssen u.a. dasselbe Adressierungsschema haben
    - ➔ z.B. Ethernet und Token-Ring ist möglich, Ethernet und ATM nicht
  - ➔ Skalierbarkeit:
    - ➔ Broadcasts
    - ➔ *Spanning-Tree-Algorithmus* (☞ 4.4)
  - ➔ Transparenz:
    - ➔ Latenz
    - ➔ Verlust von Frames bei Überlast

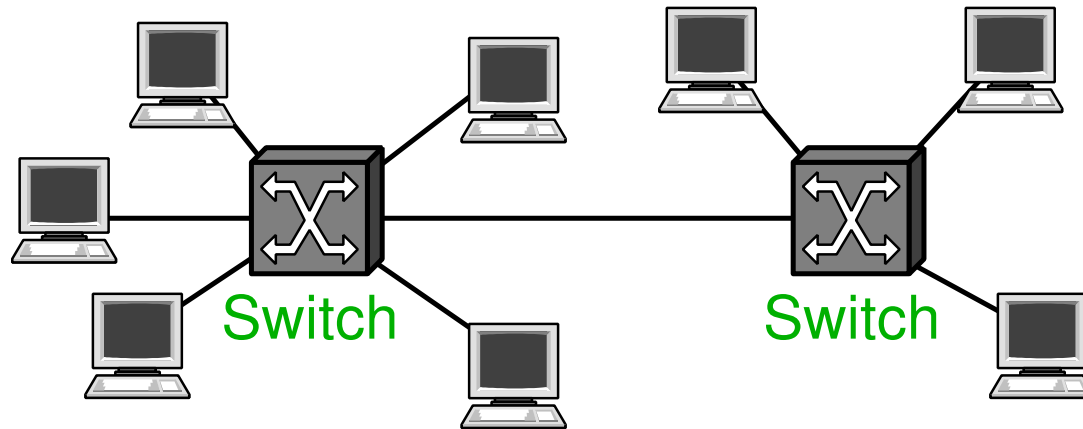
## 4.2 Switching: Einführung ...



- ➔ **LAN Switch:** Vermittlungsknoten auf der Sicherungsschicht
  - ➔ kann Zieladresse im Sicherungsschicht-Header analysieren
  - ➔ gibt Frames nur an die Ports weiter, wo es notwendig ist
- ➔ **Bridge:** Switch mit 2 Ports
  - ➔ Einsatz bei 10Base5 zur Überwindung von physikalischen Beschränkungen (Leitungslänge, Anzahl Repeater, ...)

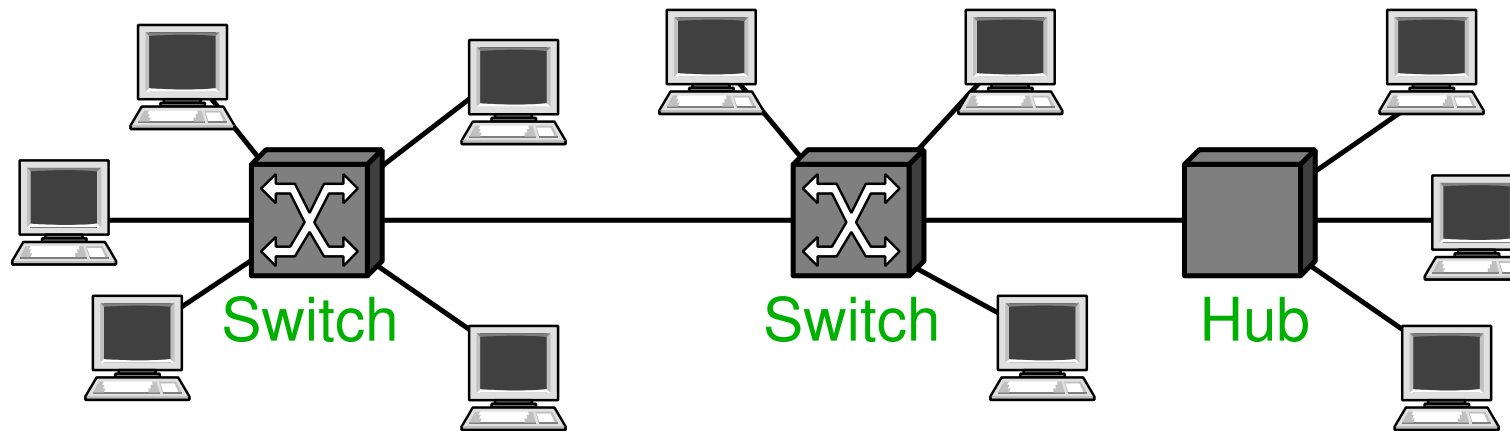


### Ein (Ethernet-)LAN mit Switches



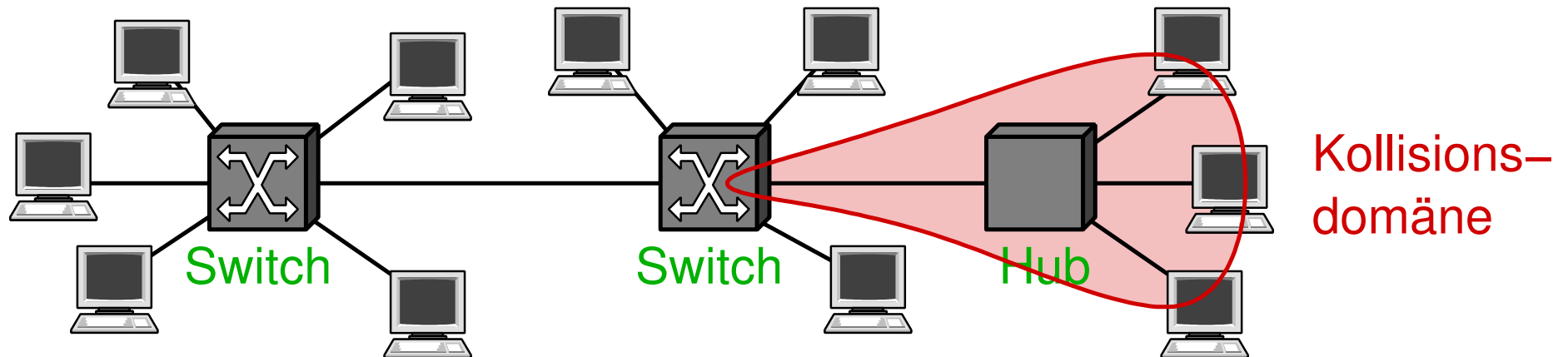
- ➔ Verwendung von Switches ist bei Ethernet ab 100 Mb/s üblich
- ➔ ermöglicht Vollduplex-Betrieb
  - ➔ in diesem Fall treten keine Kollisionen mehr auf
  - ➔ aber: Switch muß ggf. Frames zwischenspeichern

### Ein (Ethernet-)LAN mit Switches



- ➔ Verwendung von Switches ist bei Ethernet ab 100 Mb/s üblich
  - ➔ ermöglicht Vollduplex-Betrieb
    - ➔ in diesem Fall treten keine Kollisionen mehr auf
    - ➔ aber: Switch muß ggf. Frames zwischenspeichern
- ➔ Möglich auch: gemischter Betrieb mit Switches und Hubs
  - ➔ Auswirkung von Kollisionen sind auf den Bereich des Hubs eingeschränkt (**Kollisionsdomäne**)

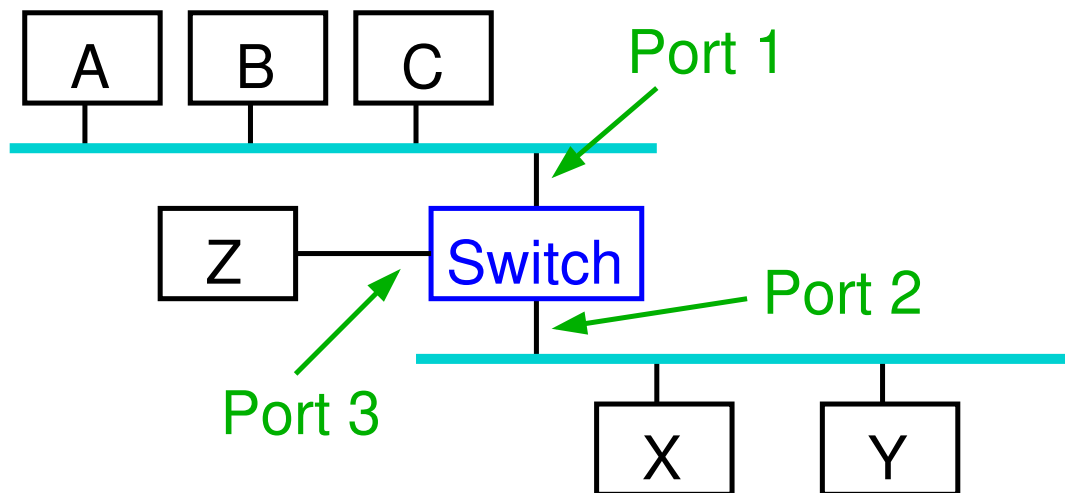
### Ein (Ethernet-)LAN mit Switches



- ➔ Verwendung von Switches ist bei Ethernet ab 100 Mb/s üblich
  - ➔ ermöglicht Vollduplex-Betrieb
    - ➔ in diesem Fall treten keine Kollisionen mehr auf
    - ➔ aber: Switch muß ggf. Frames zwischenspeichern
- ➔ Möglich auch: gemischter Betrieb mit Switches und Hubs
  - ➔ Auswirkung von Kollisionen sind auf den Bereich des Hubs eingeschränkt (**Kollisionsdomäne**)

### Automatisches Erstellen der Weiterleitungstabelle

➔ Beispiel:



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	3

- ➔ Switch untersucht die Quelladresse jedes eingehenden Frames
  - ➔ falls nötig, Erzeugung bzw. Aktualisierung eines Tabelleneintrags
- ➔ Eintrag für eine Adresse wird gelöscht, wenn längere Zeit kein Frame mit dieser Quelladresse ankommt



### Verhalten beim Eintreffen eines Frames

- ➔ Quell- und Ziel-Port identisch:
  - ➔ Frame verwerfen
- ➔ Quell- und Ziel-Port verschieden:
  - ➔ Frame an Ziel-Port weiterleiten
- ➔ Ziel-Port unbekannt:
  - ➔ weiterleiten an alle Ports (außer den Empfangsport)
- ➔ Broadcast-Frames werden immer an alle Ports geleitet
- ➔ Weiterleitungstabelle kann begrenzte Größe haben
  - ➔ Vollständigkeit ist nicht notwendig
  - ➔ Tabelle dient nur als Cache für aktive Knoten

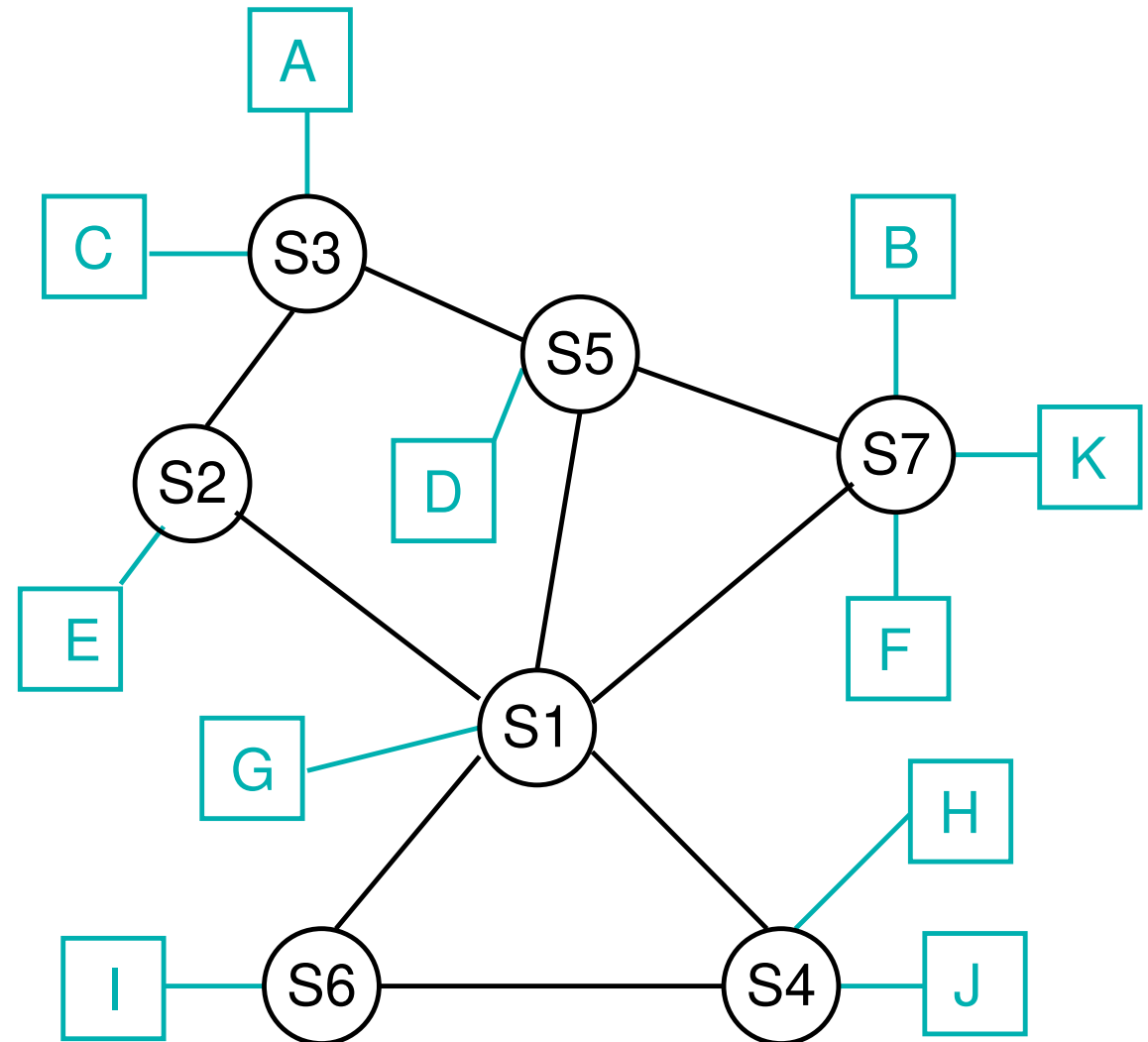


## 4.4 Spanning-Tree-Algorithmus



### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)

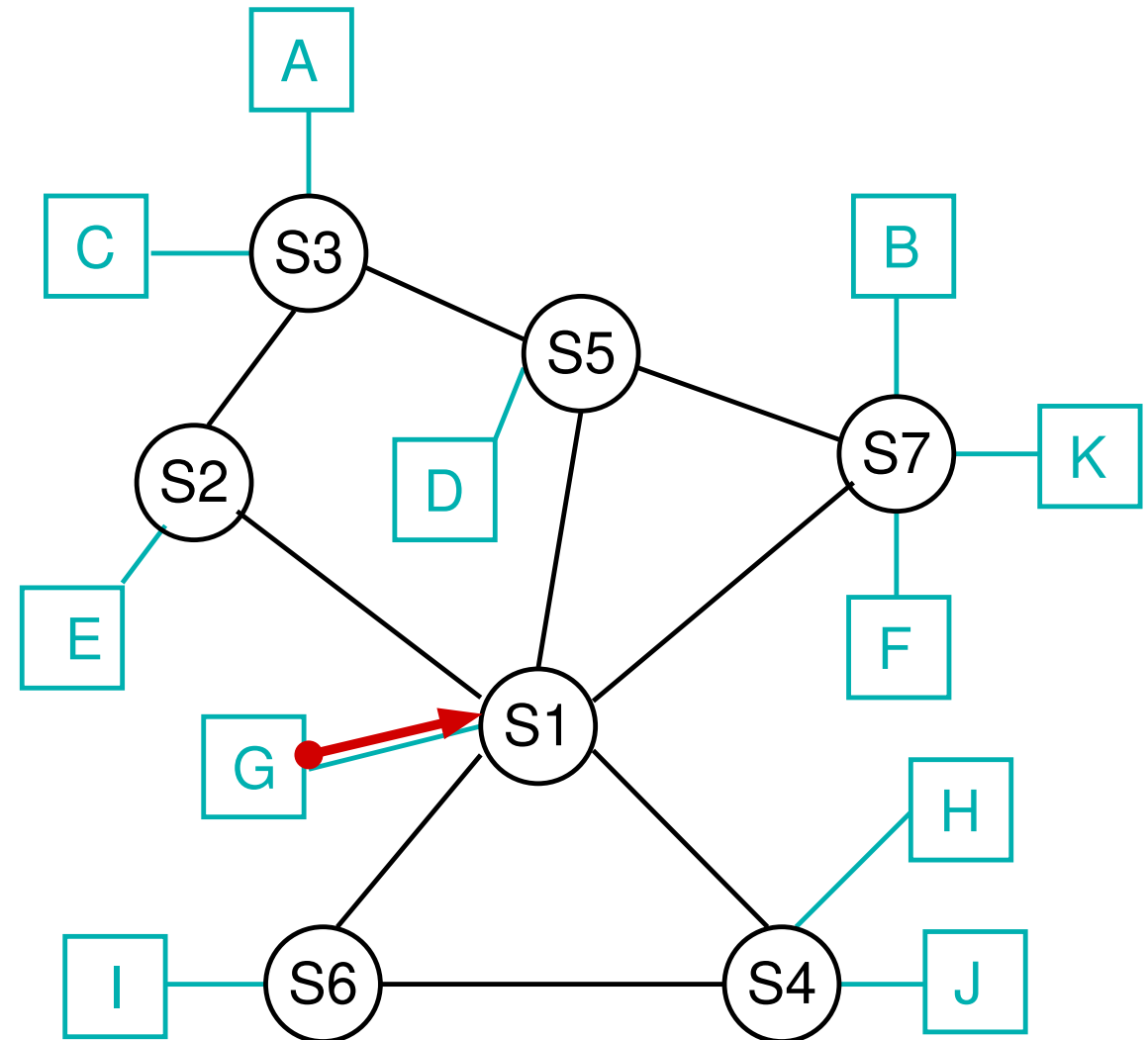


## 4.4 Spanning-Tree-Algorithmus



### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)

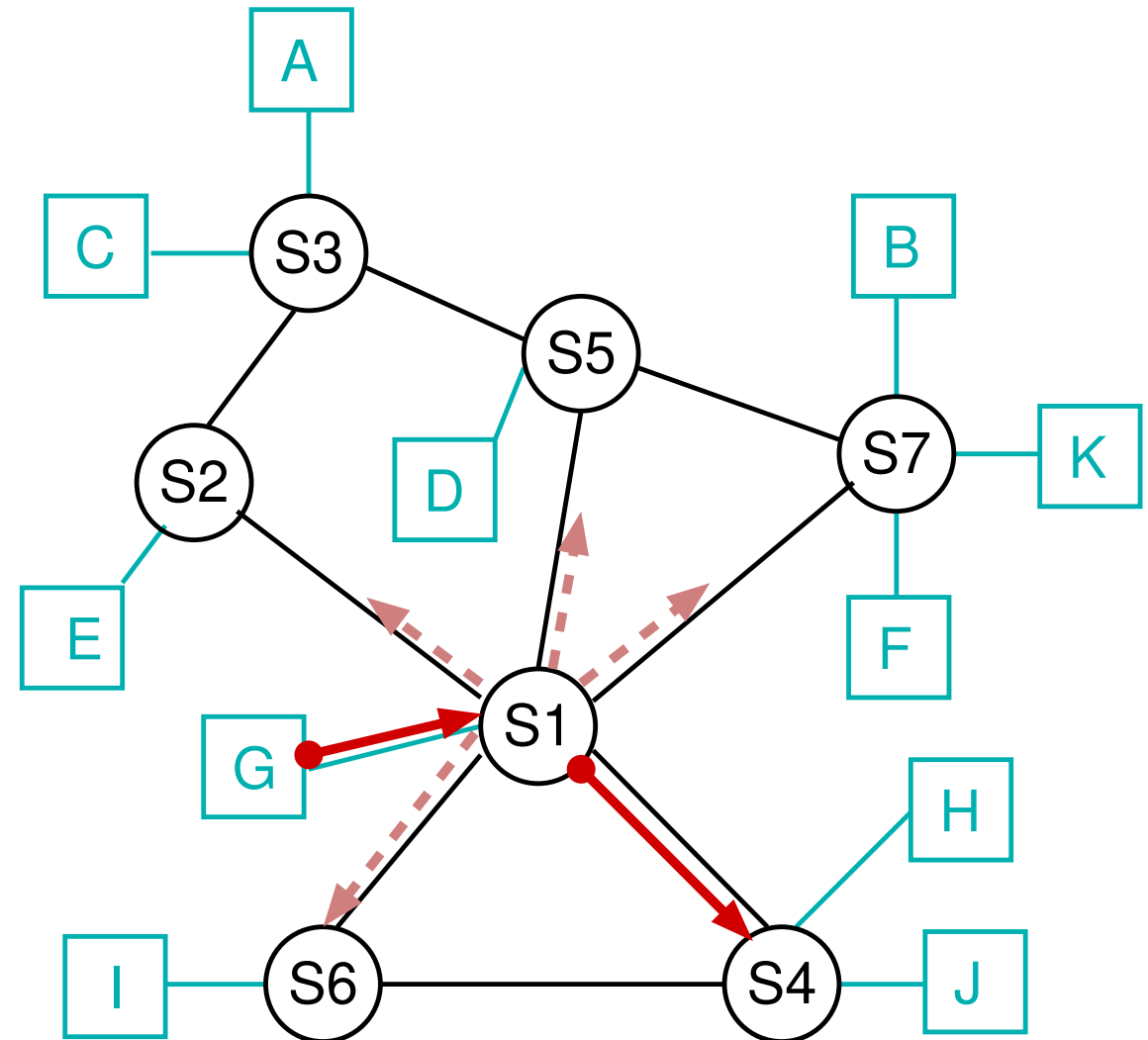


## 4.4 Spanning-Tree-Algorithmus



### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)

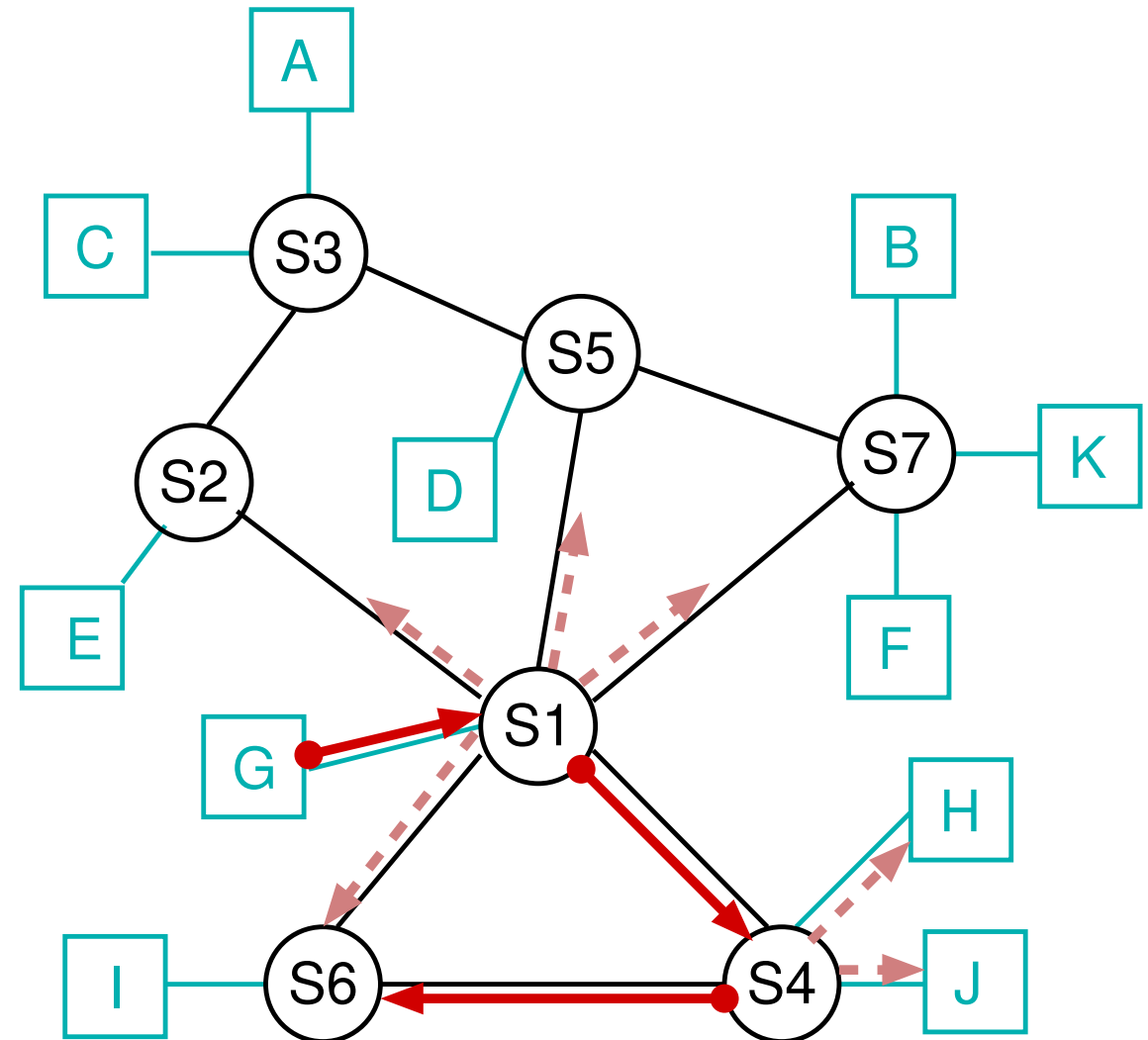


## 4.4 Spanning-Tree-Algorithmus



### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)

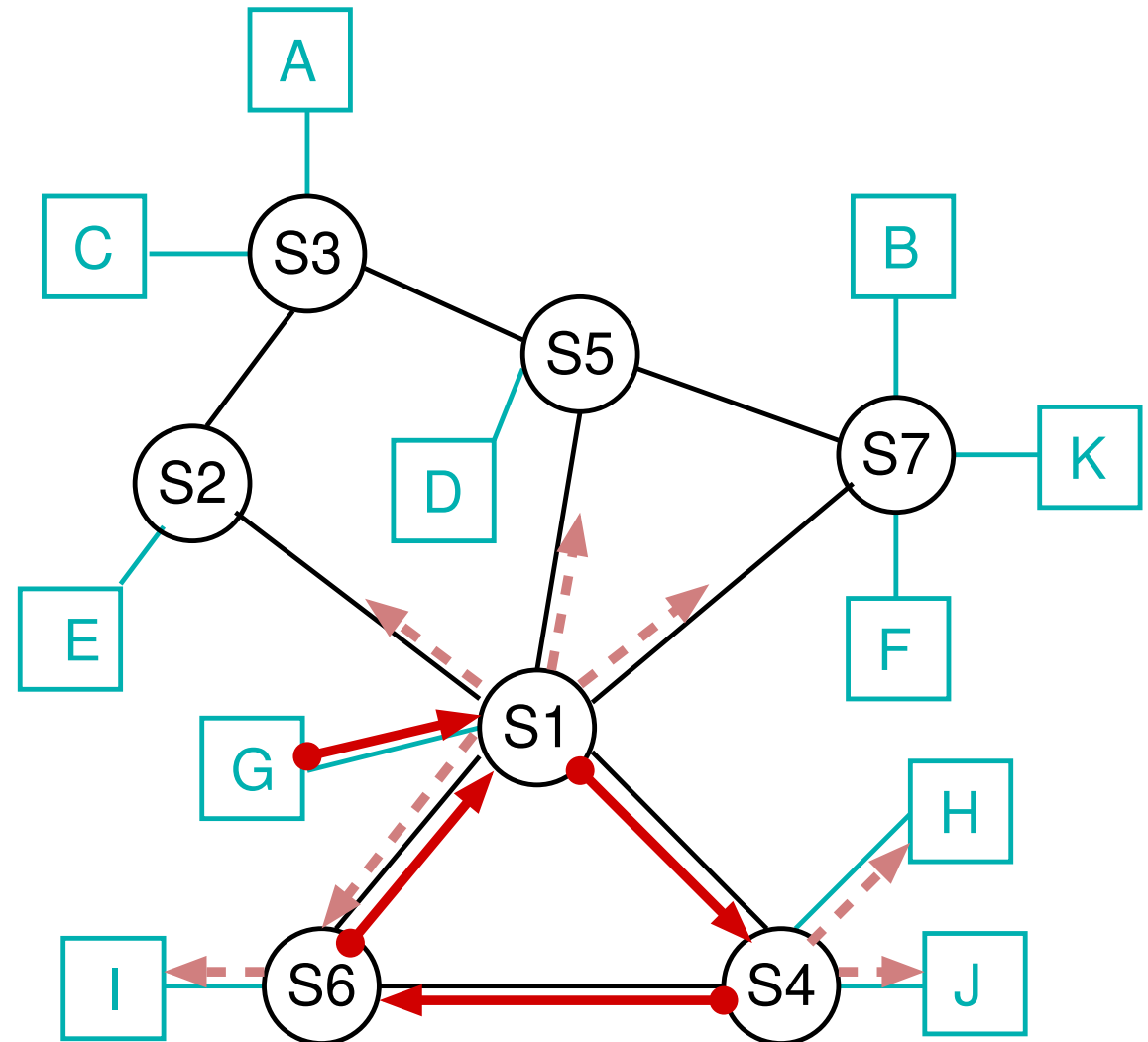


## 4.4 Spanning-Tree-Algorithmus



### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)



## 4.4 *Spanning-Tree-Algorithmus*

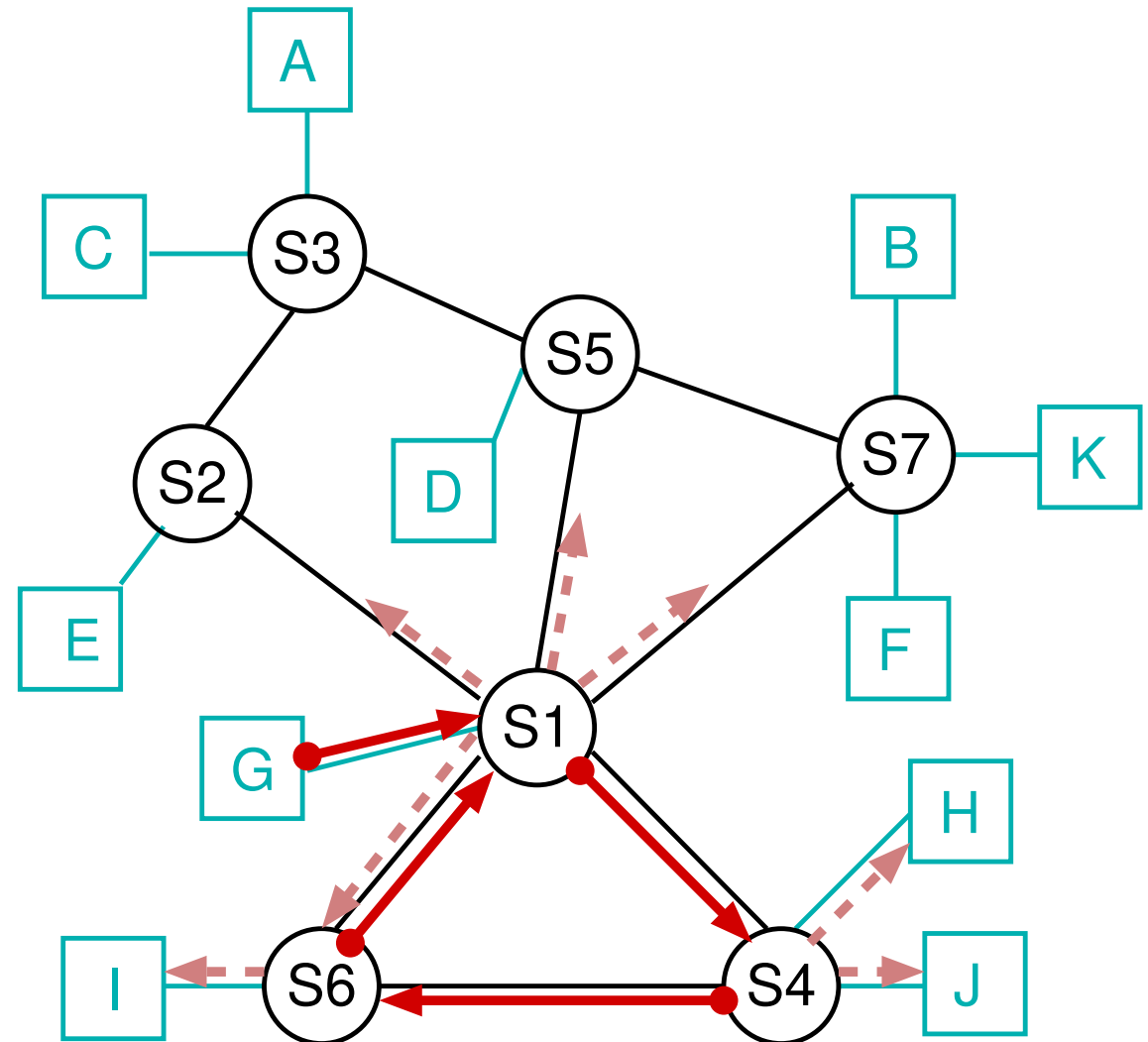


### Problem bei lernenden Switches: Zyklen

- ➔ Z.B. S1 – S4 – S6
- ➔ Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)

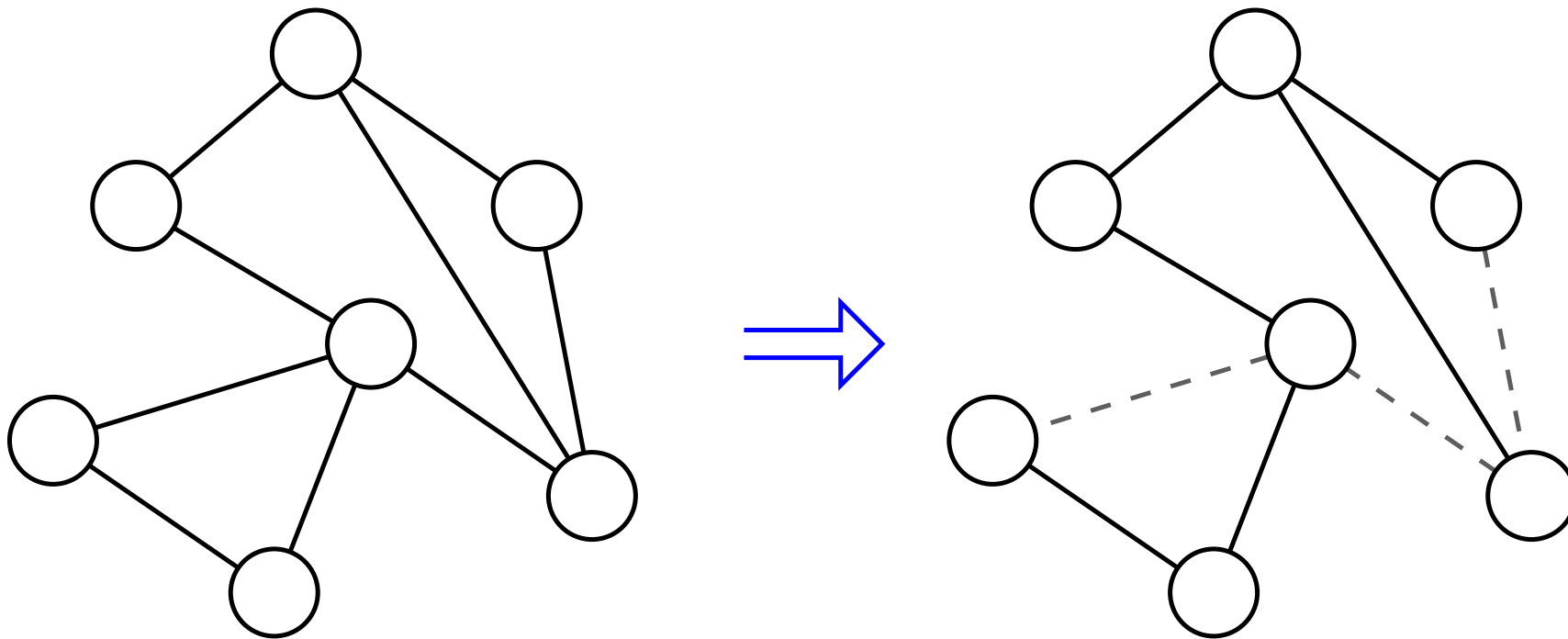
### Abhilfe: *Spanning-Tree-Algorithmus*

- ➔ Reduziert das Netzwerk auf einen zyklensfreien Graphen (Baum)
- ➔ Einige Ports der Switches werden deaktiviert



### Aufspannender Baum

➔ Zyklischer Graph und aufspannender Baum:



➔ Der aufspannende Baum ist nicht eindeutig



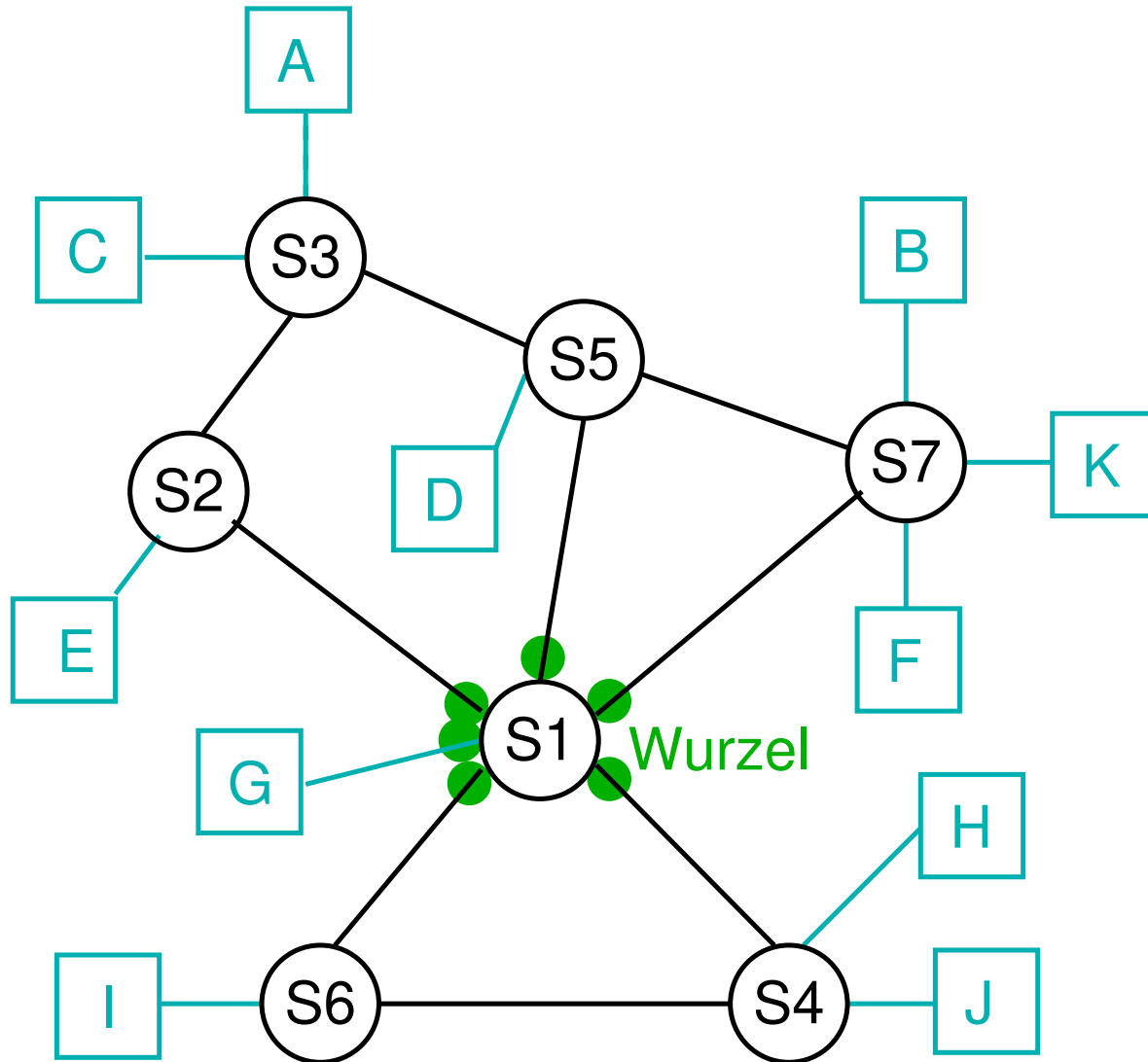
### Idee des Algorithmus

- ➔ Jeder Switch hat eine eindeutige Kennung (z.B. S1, S2)
- ➔ Wähle den Switch mit der kleinsten Kennung als Wurzel
- ➔ Jeder Switch bestimmt seinen **Root Port**
  - ➔ der Port mit der geringsten Entfernung zur Wurzel
  - ➔ bei Gleichheit entscheidet Port-Priorität bzw. Port-Nummer
- ➔ Wähle für jedes LAN-Segment den Port des Switches mit der geringsten Entfernung zur Wurzel als **Designated Port**
  - ➔ bei Gleichheit entscheidet die Switch-Kennung
- ➔ Alle anderen Ports sind **Non-Designated Ports**
  - ➔ diese Ports sind blockiert (*blocking*) und leiten keine Frames weiter





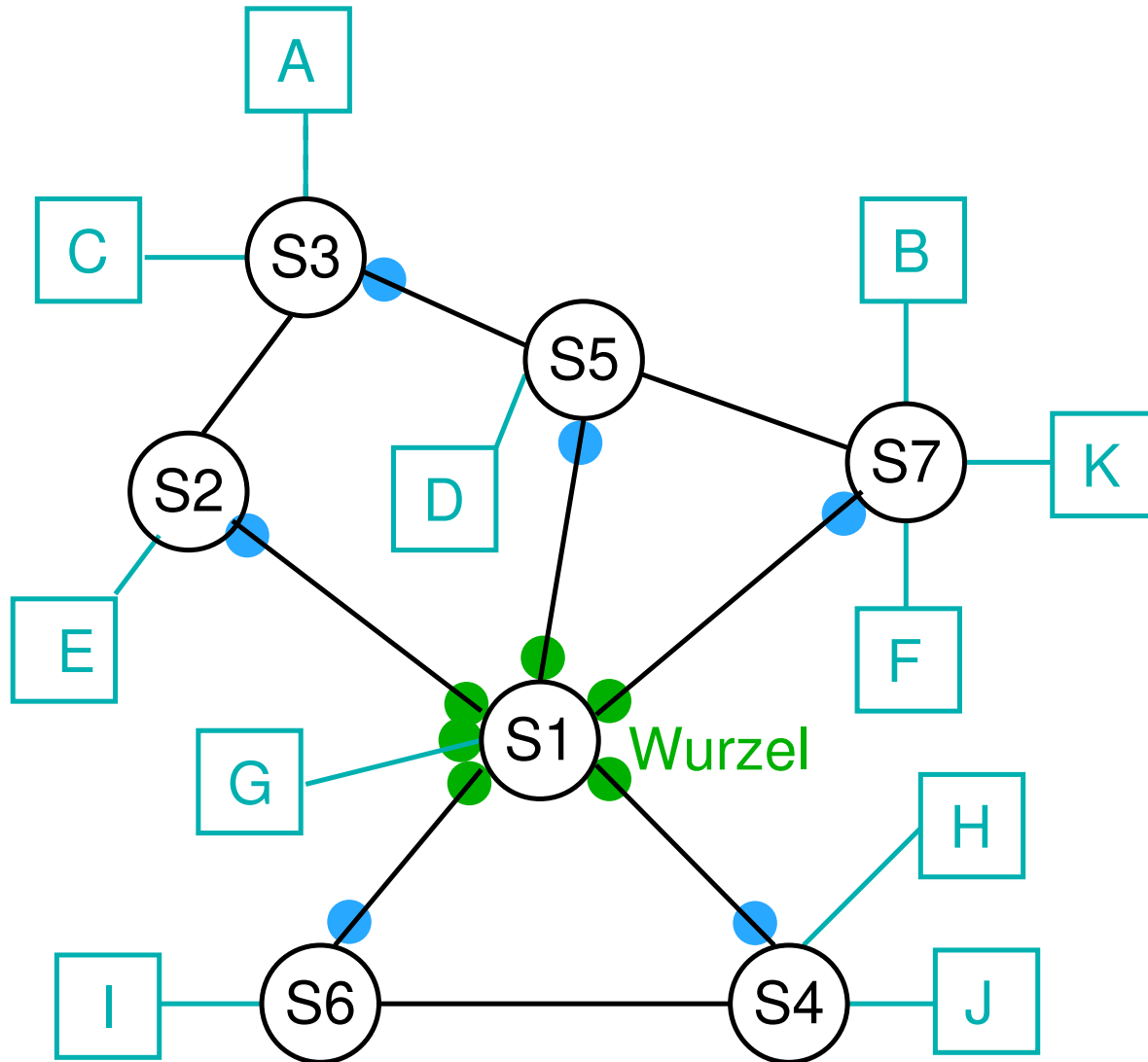
## Aufspannender Baum des Beispielnetzes



- Root Port
- Non-Designated Port
- Designated Port



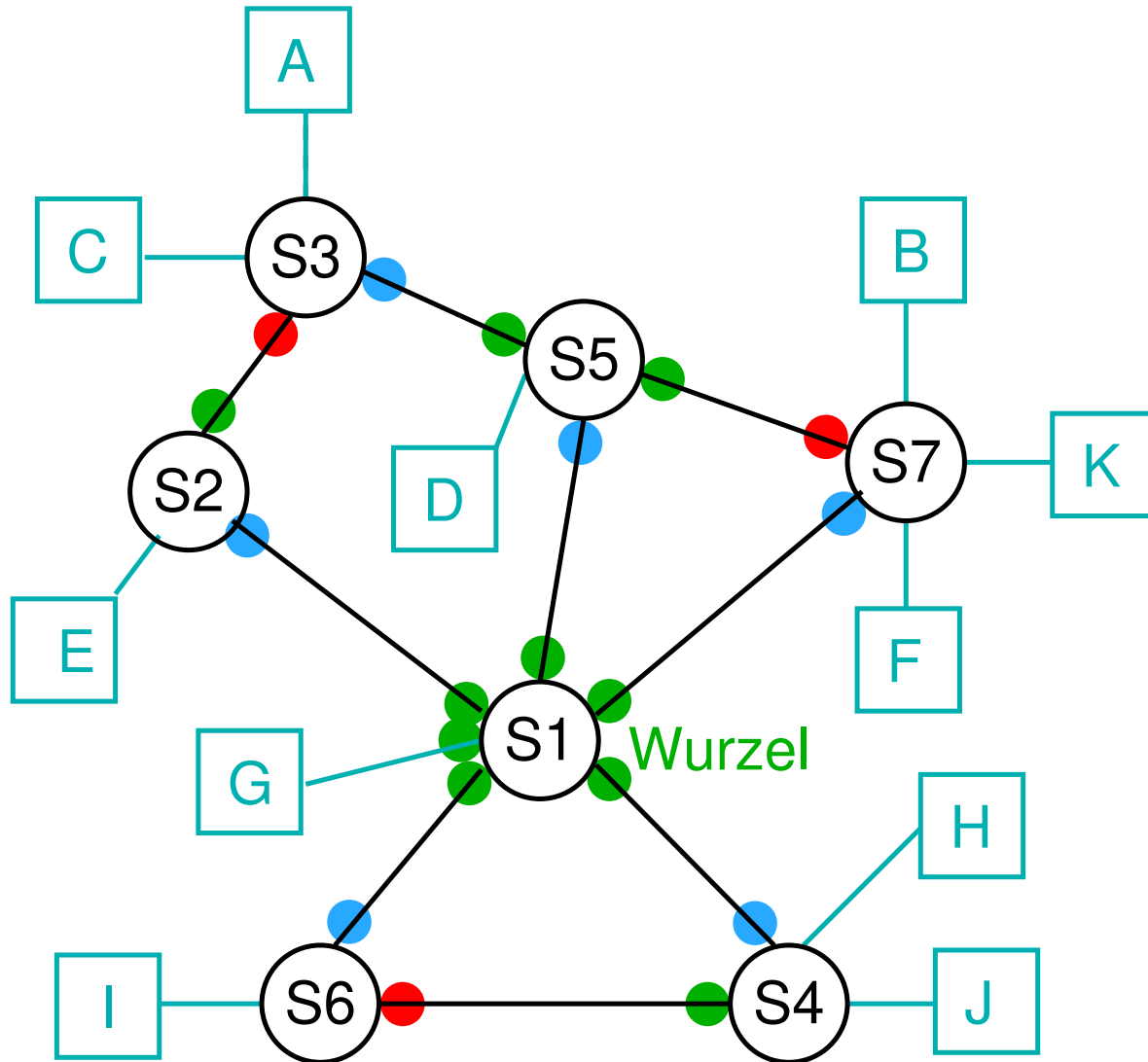
## Aufspannender Baum des Beispielnetzes



- Root Port
- Non-Designated Port
- Designated Port



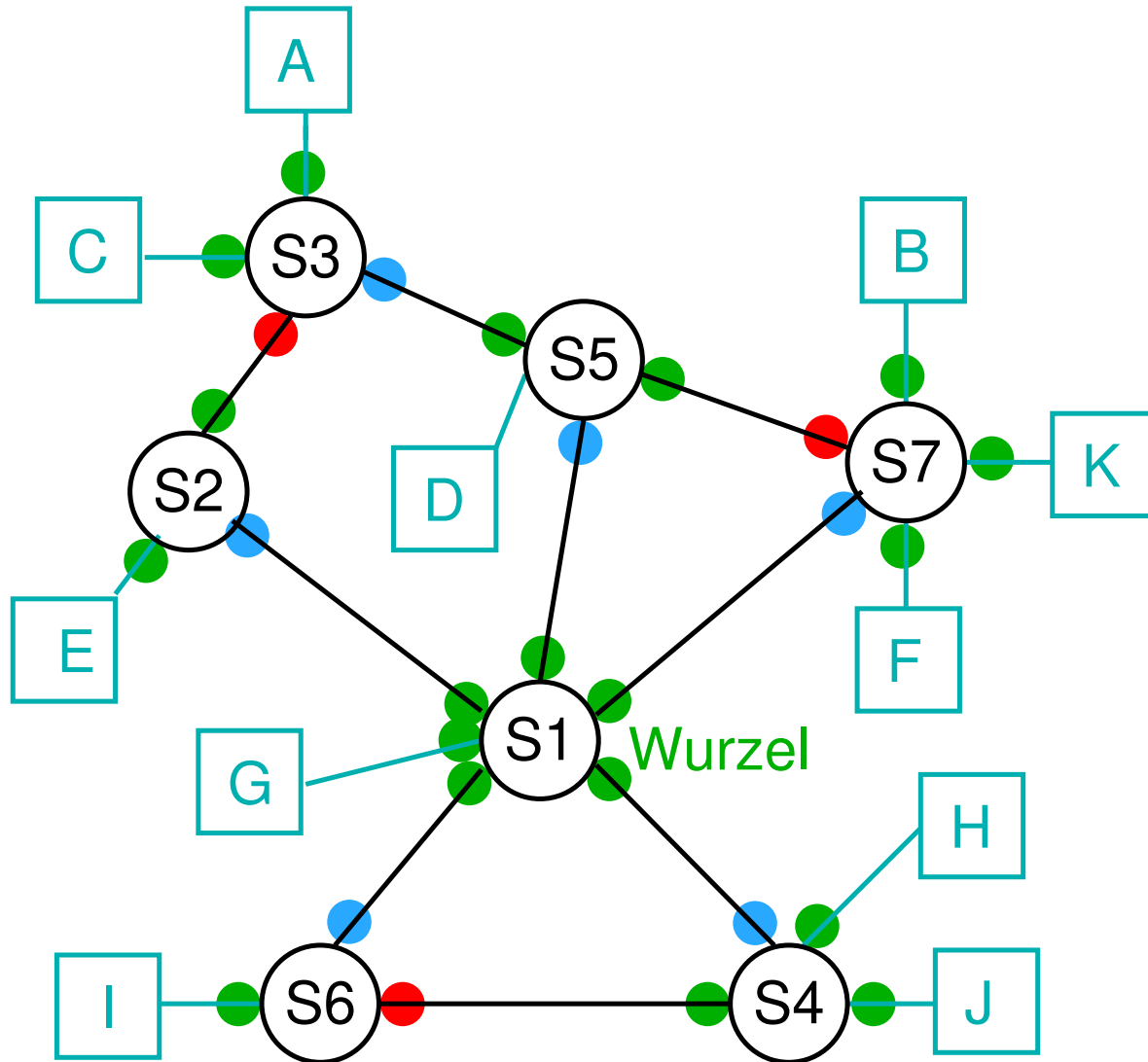
## Aufspannender Baum des Beispielnetzes



- Root Port
- Non-Designated Port
- Designated Port



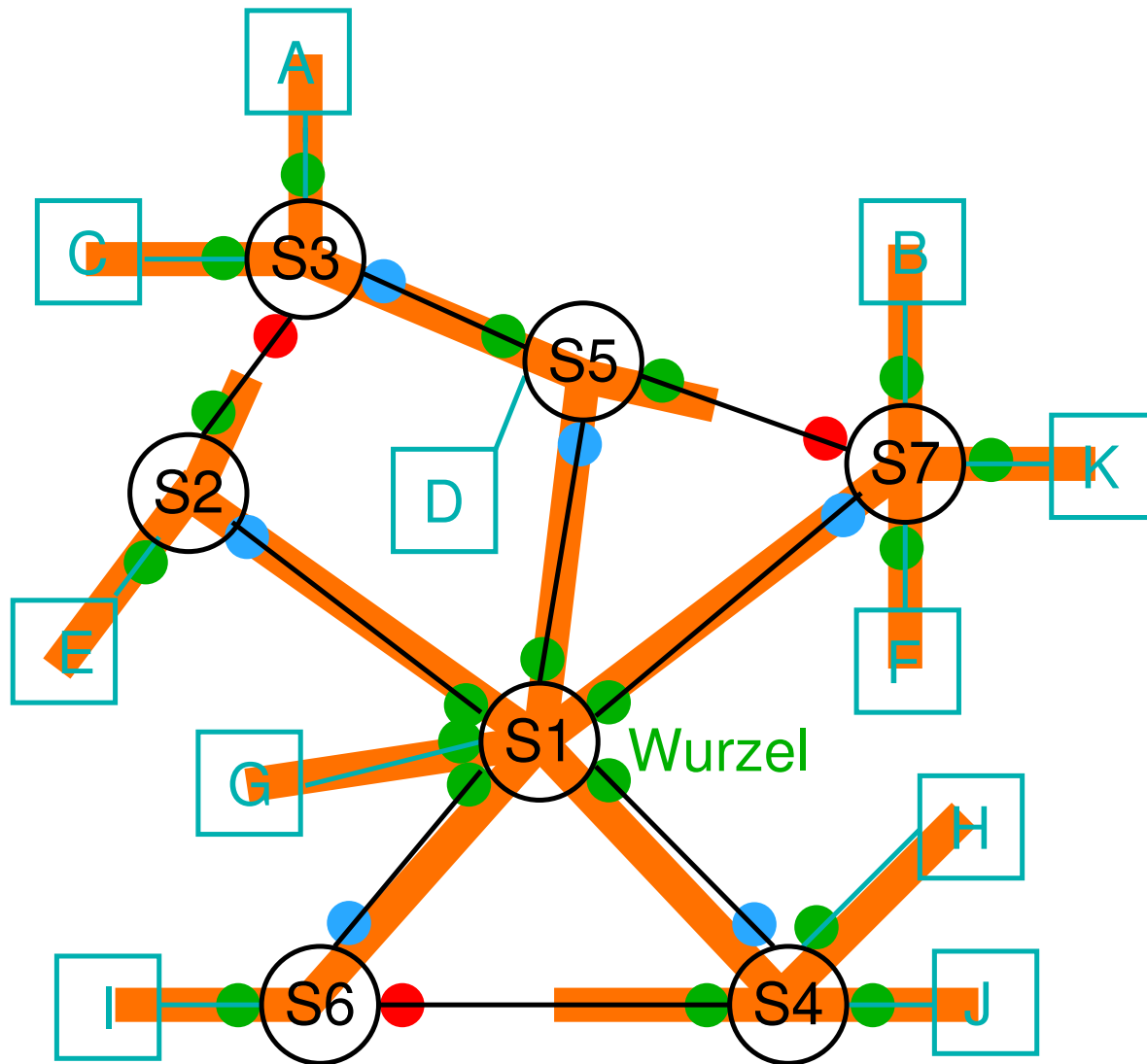
## Aufspannender Baum des Beispielnetzes



- Root Port
- Non-Designated Port
- Designated Port



## Aufspannender Baum des Beispielnetzes





### *Spanning-Tree-Protokoll (STP)*

- ➔ Die Knoten tauschen Konfigurations-Nachrichten aus:
  - ➔ Kennung des sendenden Switches
  - ➔ vermutete Wurzel-Kennung
  - ➔ eigene Entfernung zu dieser Wurzel
- ➔ Jeder Switch behält die beste Nachricht. Besser heißt dabei:
  - ➔ Wurzel-Kennung kleiner, oder
  - ➔ Wurzel-Kennung gleich, Entfernung kleiner, oder
  - ➔ Wurzel-Kennung und Entfernung gleich, Sender-Kennung kleiner
- ➔ Jeder Switch startet als Wurzel, bis dies widerlegt ist

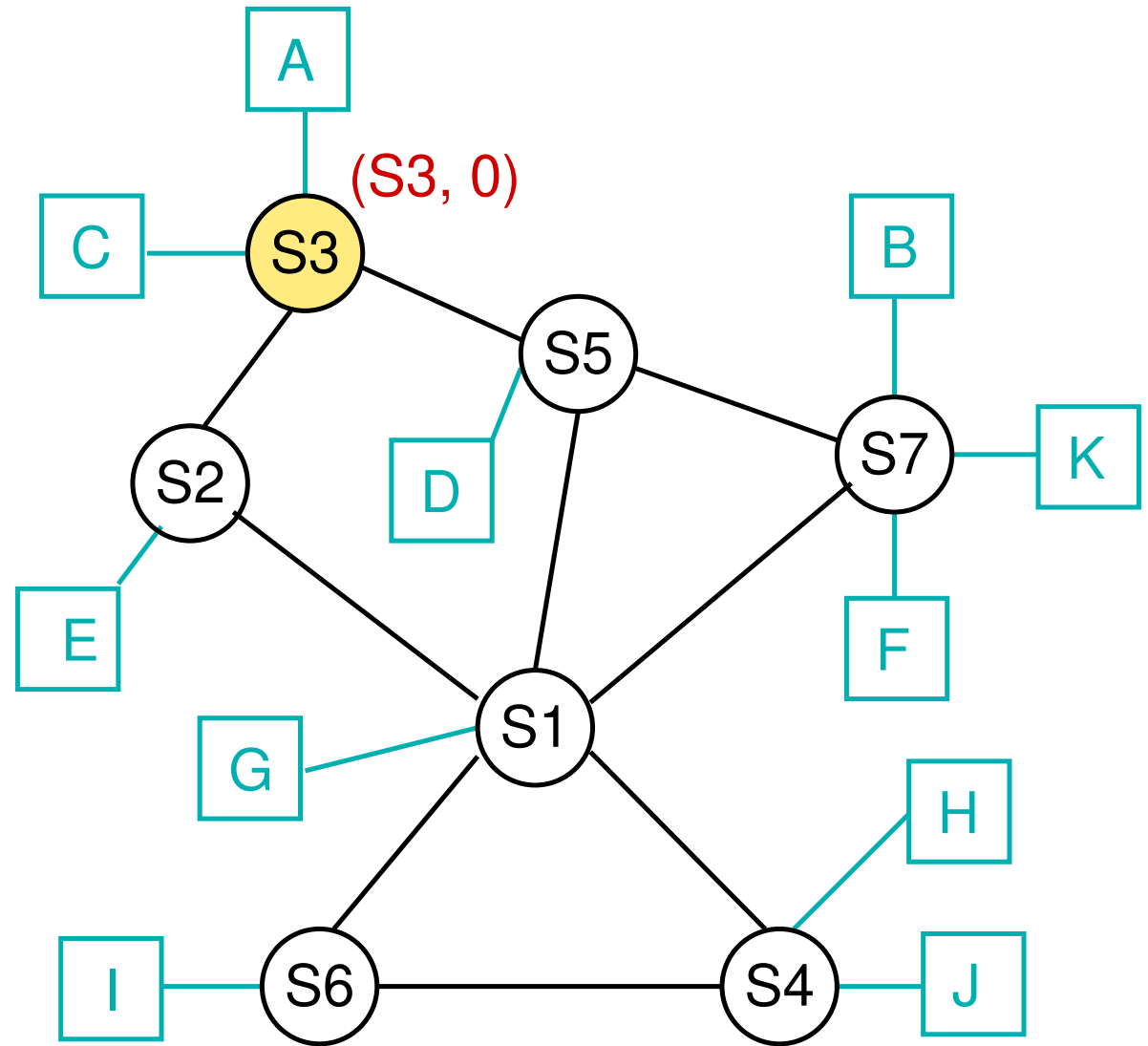


### *Spanning-Tree-Protokoll (STP)* ...

- ➔ Wenn ein Switch erfährt, daß er nicht die Wurzel ist, stellt er das Generieren von Nachrichten ein, leitet Nachrichten aber nach wie vor weiter
  - ➔ am Ende erzeugt nur noch die Wurzel Nachrichten
  
- ➔ Die Wurzel generiert weiter periodisch Konfigurations-Nachrichten
  - ➔ werden im Baum nur noch nach unten weitergegeben
  - ➔ automatische Rekonfiguration bei Ausfall der Wurzel
  
- ➔ Topologie-Änderungen werden von Switches zunächst an die Wurzel gesendet, diese gibt sie an alle weiter



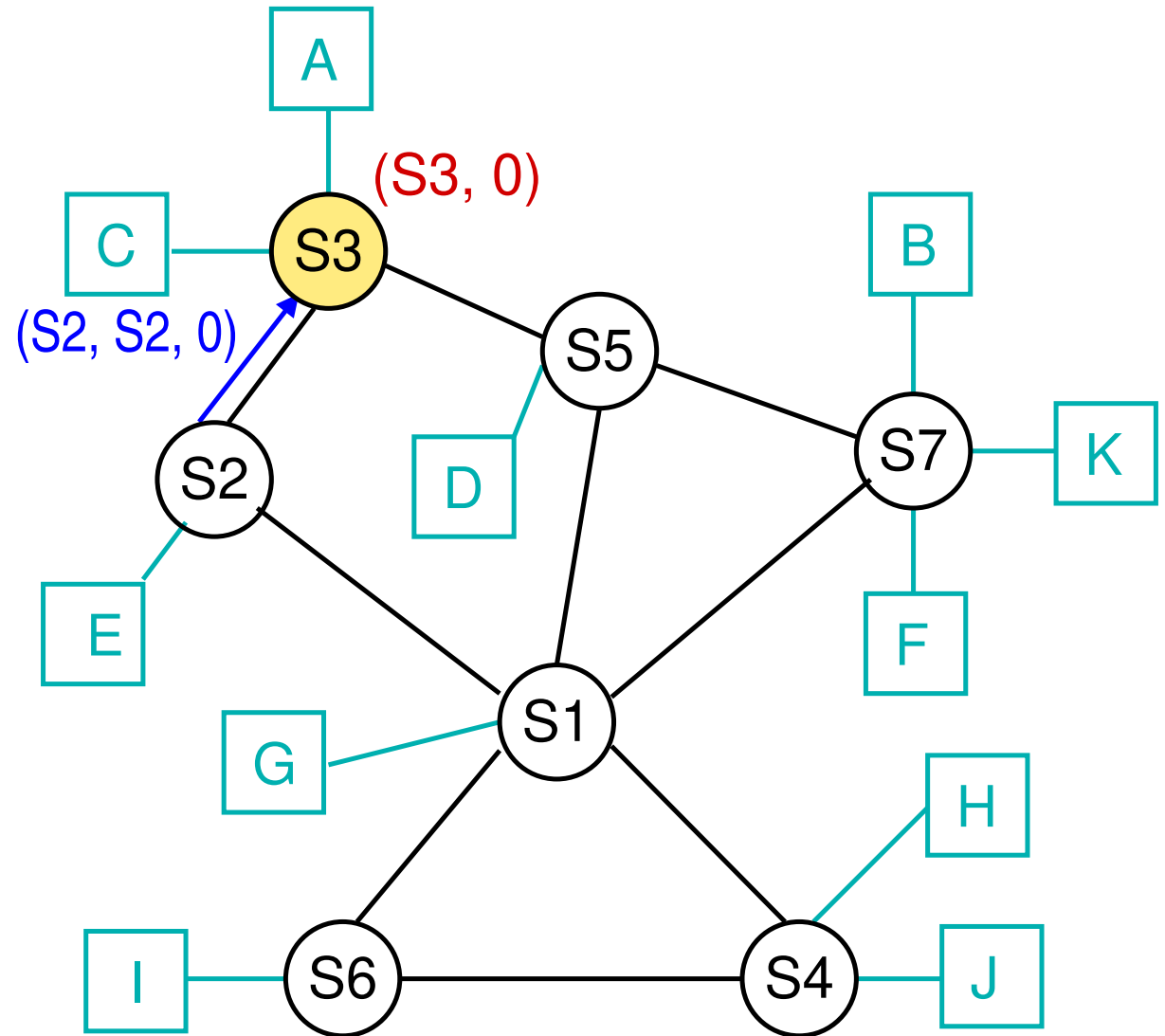
## Beispiel





## Beispiel

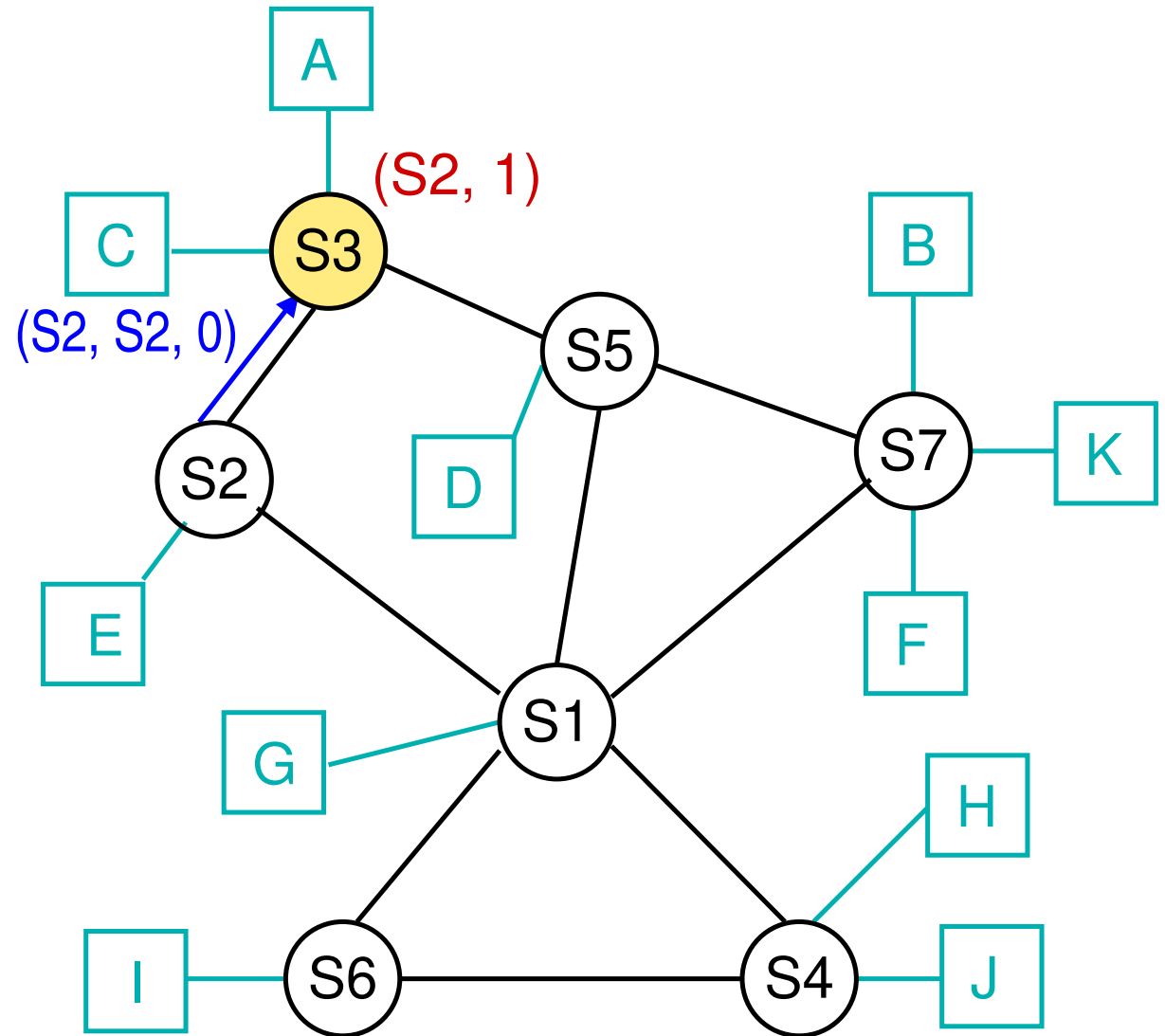
1.  $S2 \rightarrow S3$





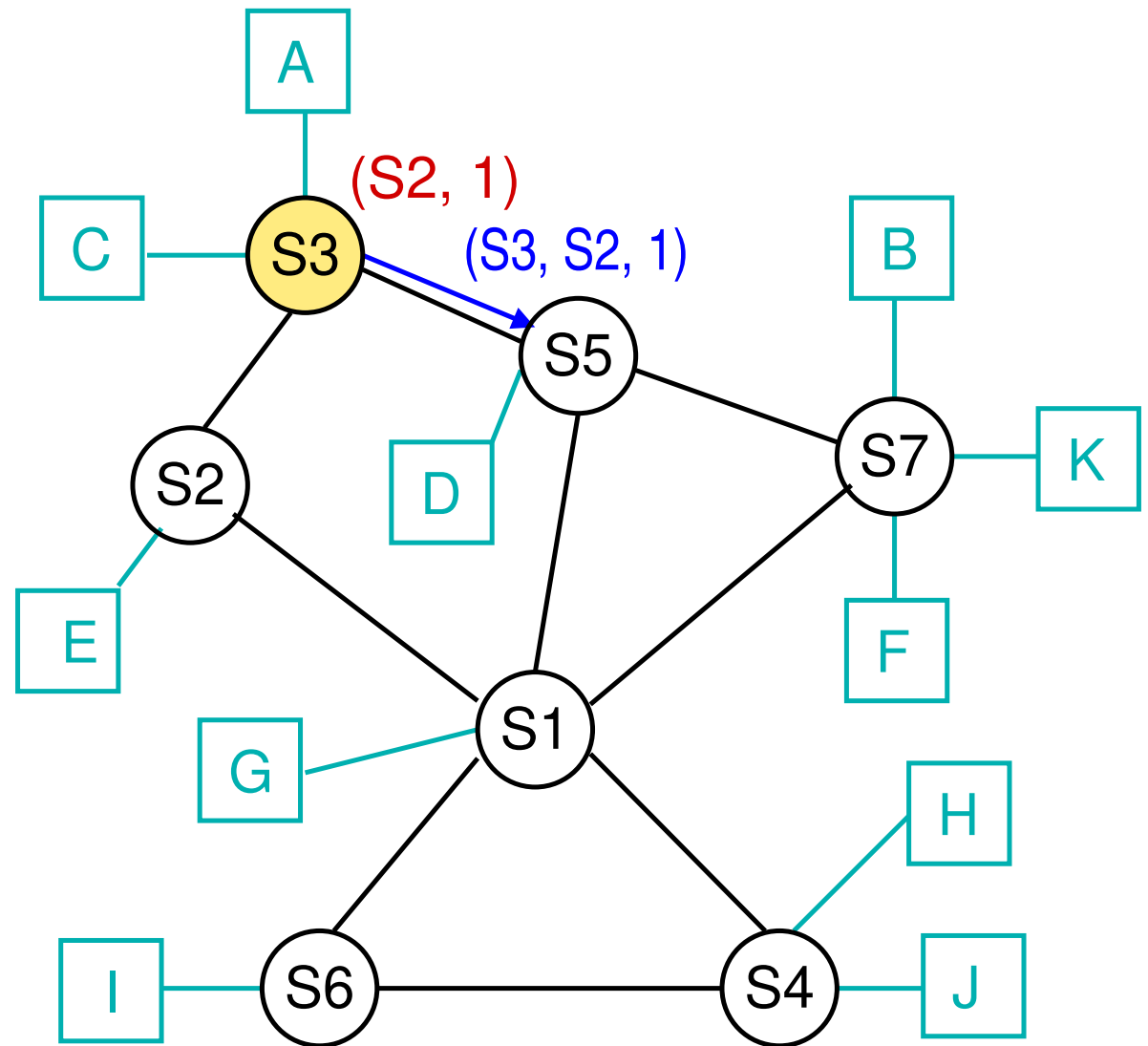
## Beispiel

1.  $S2 \rightarrow S3$
2.  $S3$ :  $S2$  ist Wurzel



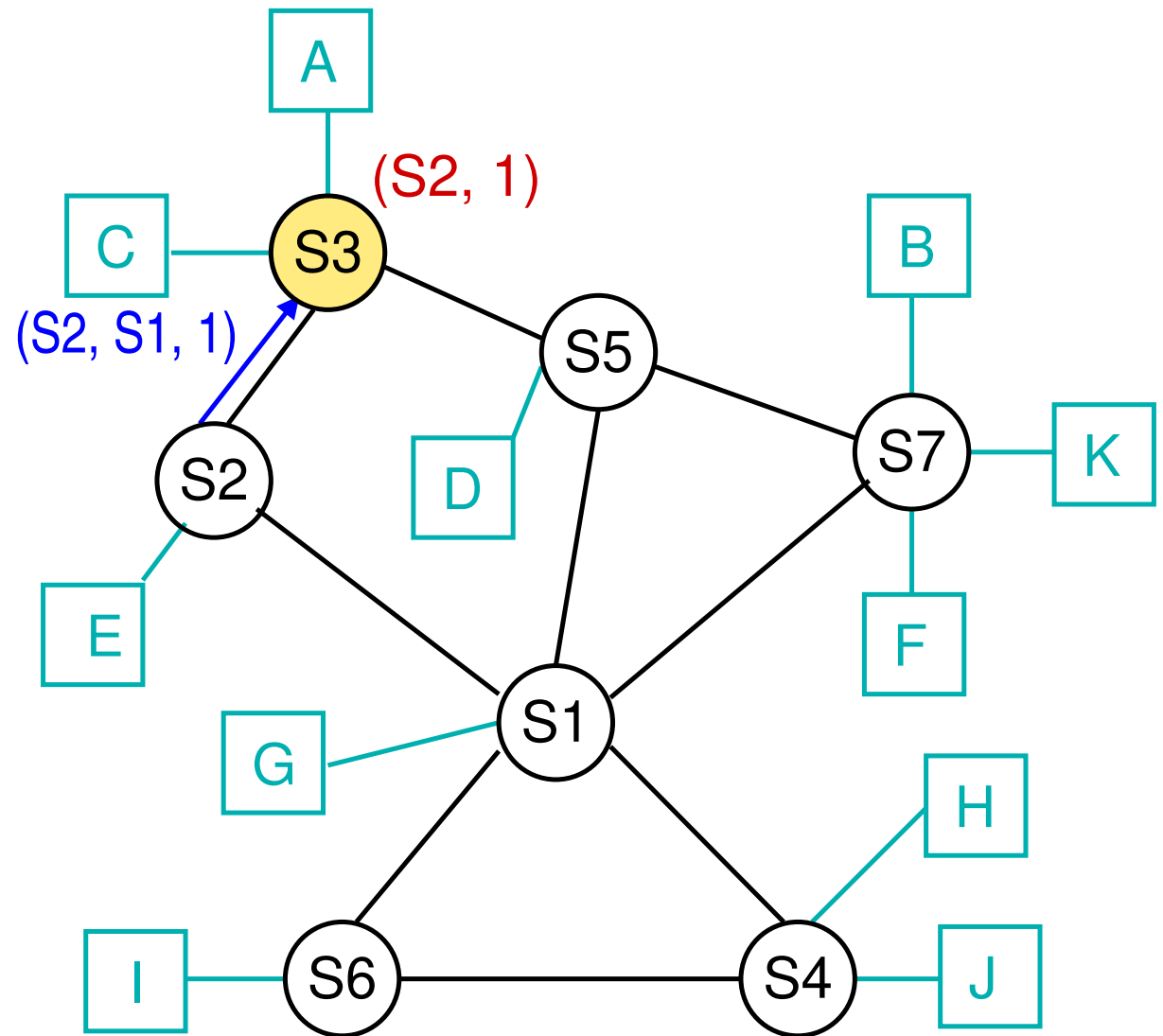
### Beispiel

1.  $S2 \rightarrow S3$
2.  $S3$ :  $S2$  ist Wurzel
3.  $S3 \rightarrow S5$



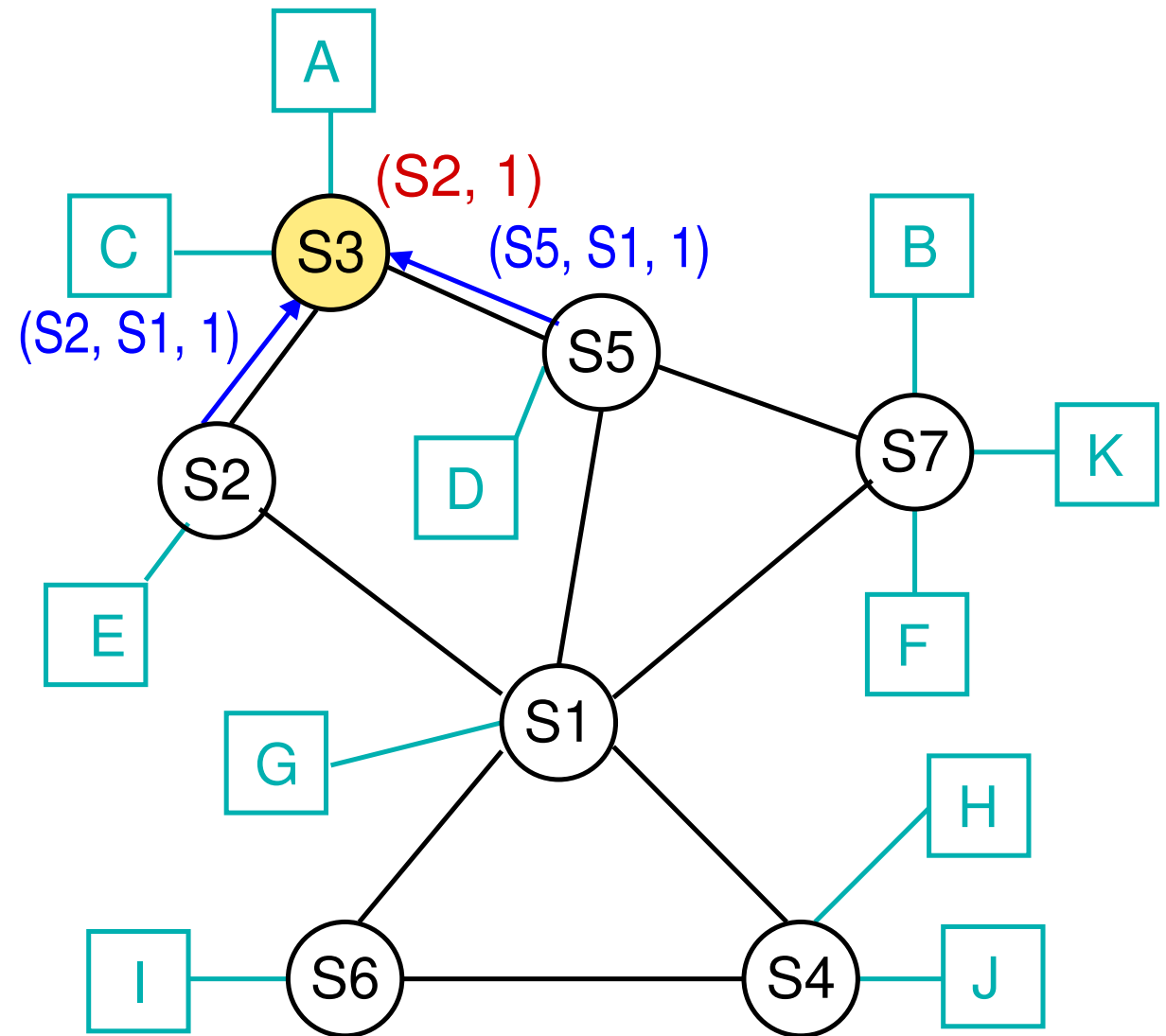
### Beispiel

1.  $S2 \rightarrow S3$
2.  $S3$ :  $S2$  ist Wurzel
3.  $S3 \rightarrow S5$
4.  $S2$ :  $S1$  ist Wurzel  
 $S2 \rightarrow S3$



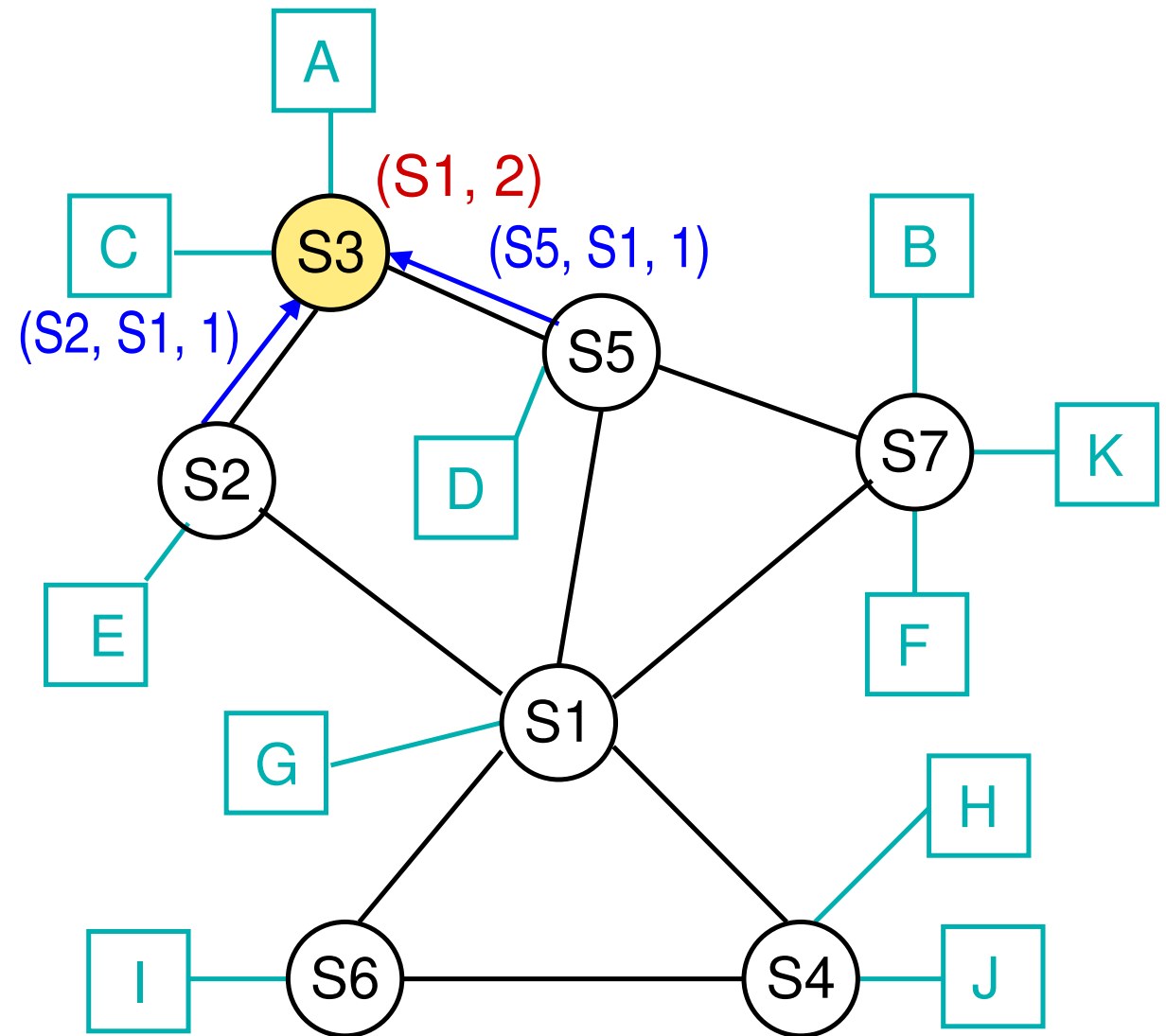
## Beispiel

1.  $S2 \rightarrow S3$
2.  $S3$ :  $S2$  ist Wurzel
3.  $S3 \rightarrow S5$
4.  $S2$ :  $S1$  ist Wurzel  
 $S2 \rightarrow S3$
5.  $S5$ :  $S1$  ist Wurzel  
 $S5 \rightarrow S3$



## Beispiel

1.  $S2 \rightarrow S3$
2.  $S3$ :  $S2$  ist Wurzel
3.  $S3 \rightarrow S5$
4.  $S2$ :  $S1$  ist Wurzel  
 $S2 \rightarrow S3$
5.  $S5$ :  $S1$  ist Wurzel  
 $S5 \rightarrow S3$
6.  $S3$ :  $S1$  ist Wurzel  
 $S2, S5$  näher an  $S1$



## 4.5 Virtuelle LANs (VLANs)



➔ Ziele:

➔ bessere Skalierung

➔ höhere Sicherheit

➔ Jedes LAN erhält einen Bezeichner (VLAN-ID)

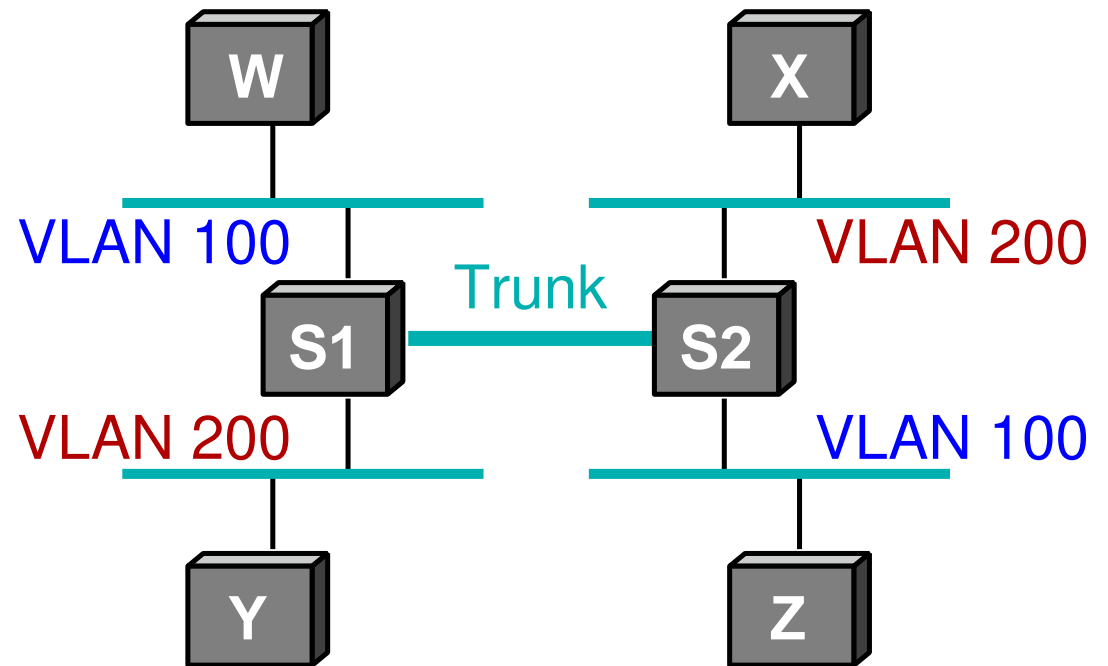
➔ Auf Trunk-Leitung: Switch fügt Header mit VLAN-ID ein bzw. entfernt ihn wieder

➔ bei Ethernet: VLAN-ID wird **in** den Frame-Header eingefügt

➔ Frames werden nur an das LAN mit der korrekten VLAN-ID weitergeleitet

➔ LANs mit verschiedenen VLAN-IDs sind logisch getrennt

➔ Kommunikation nur über Router möglich





- ➔ Weiterleitungstechniken
  - ➔ Datagrammvermittlung, virtuelle Leitungsvermittlung
- ➔ LAN-Switches
  - ➔ lernen Weiterleitungstabellen selbst
  - ➔ zur Vermeidung von Zyklen: *Spanning Tree Protokoll*
  - ➔ erlauben die Realisierung von virtuellen LANs

### Nächste Lektion:

- ➔ *Internetworking*
- ➔ Das Internet-Protokoll (IP)