

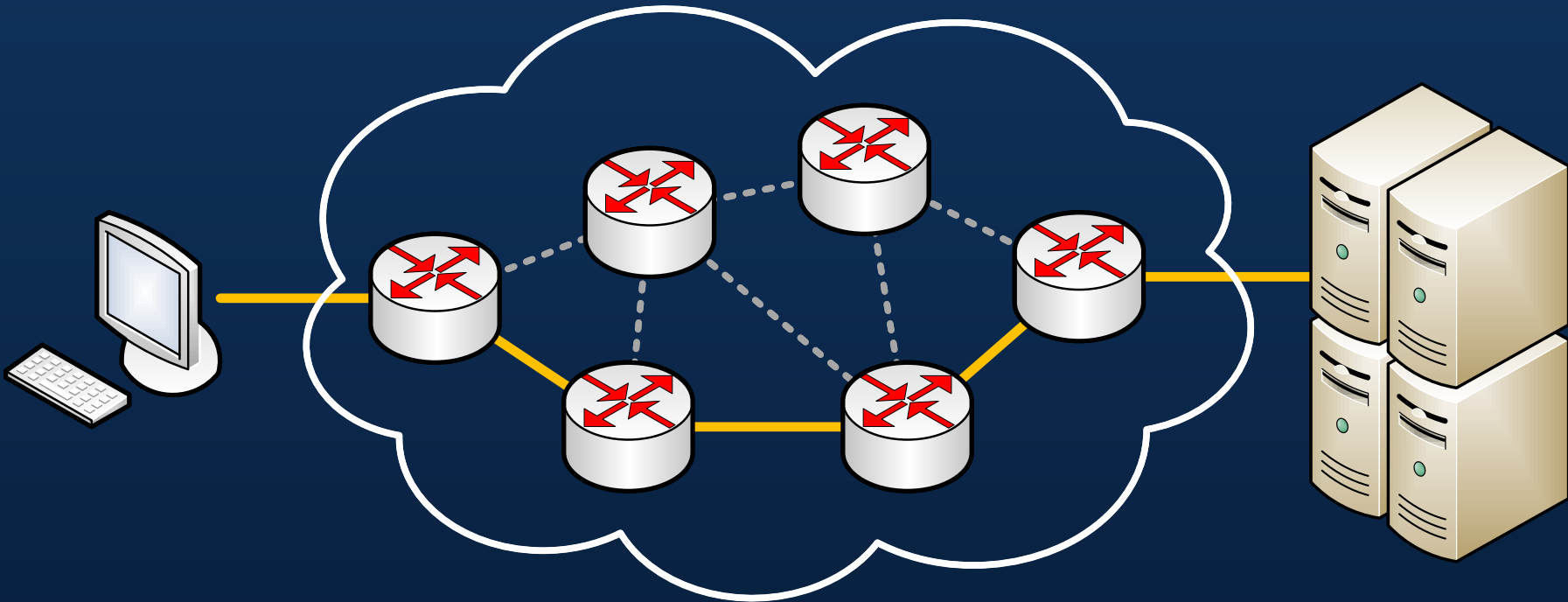


Switches und Router intern

SWITCHES und ROUTER INTERN

Einleitung / Motivation

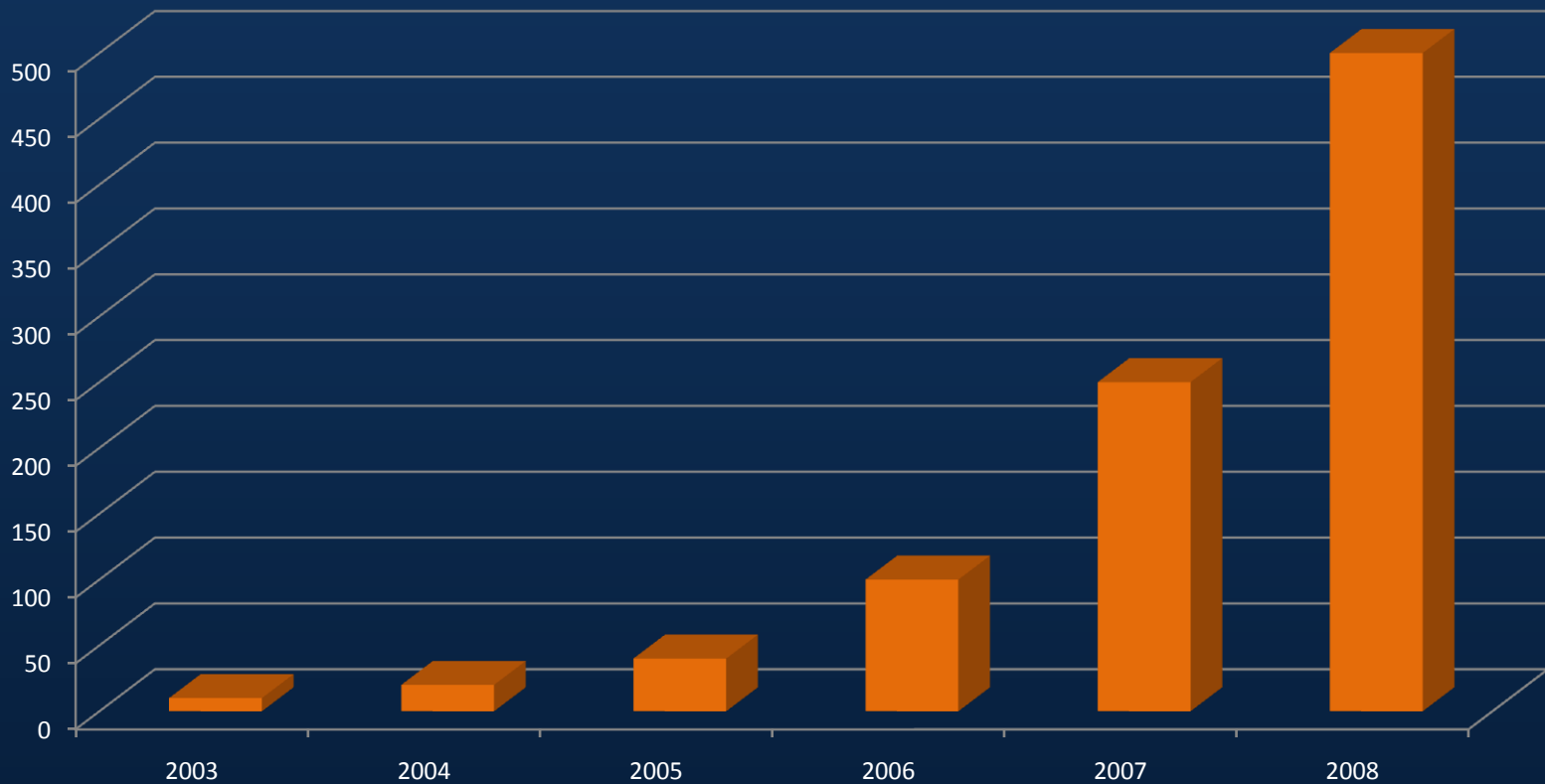
- Router und Switches bilden das Rückgrat des Internet



Einleitung / Motivation

- Fakten, Fakten, Fakten!

DE-CIX - Spitzendurchsatz in Gbit/s



Quelle: Wikipedia-Eintrag „DE-CIX“

Einleitung / Motivation

- Fakten, Fakten, Fakten!

DE-CIX - Spitzendurchsatz in Gbit/s

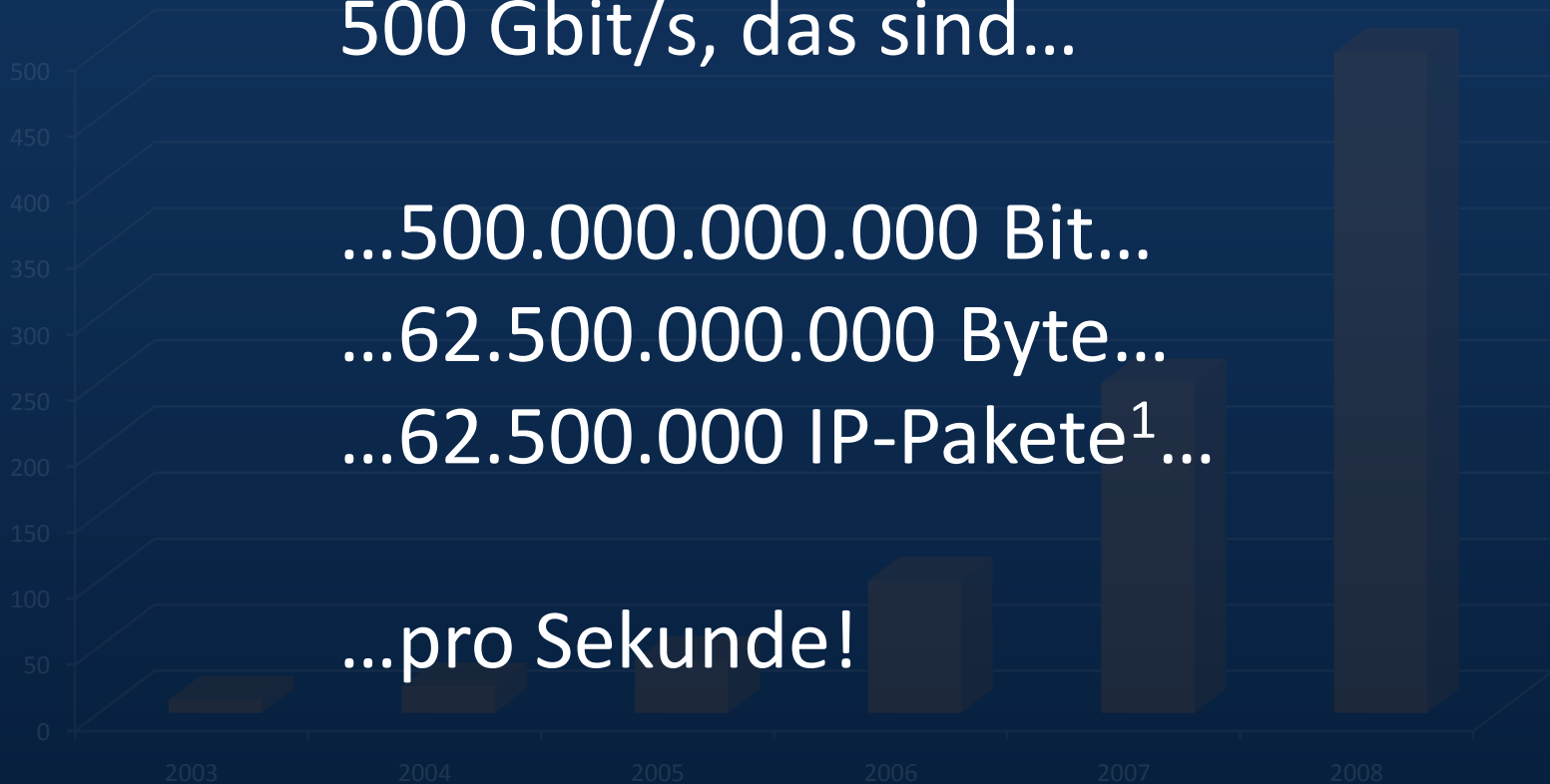
500 Gbit/s, das sind...

...500.000.000.000 Bit...

...62.500.000.000 Byte...

...62.500.000 IP-Pakete¹...

...pro Sekunde!



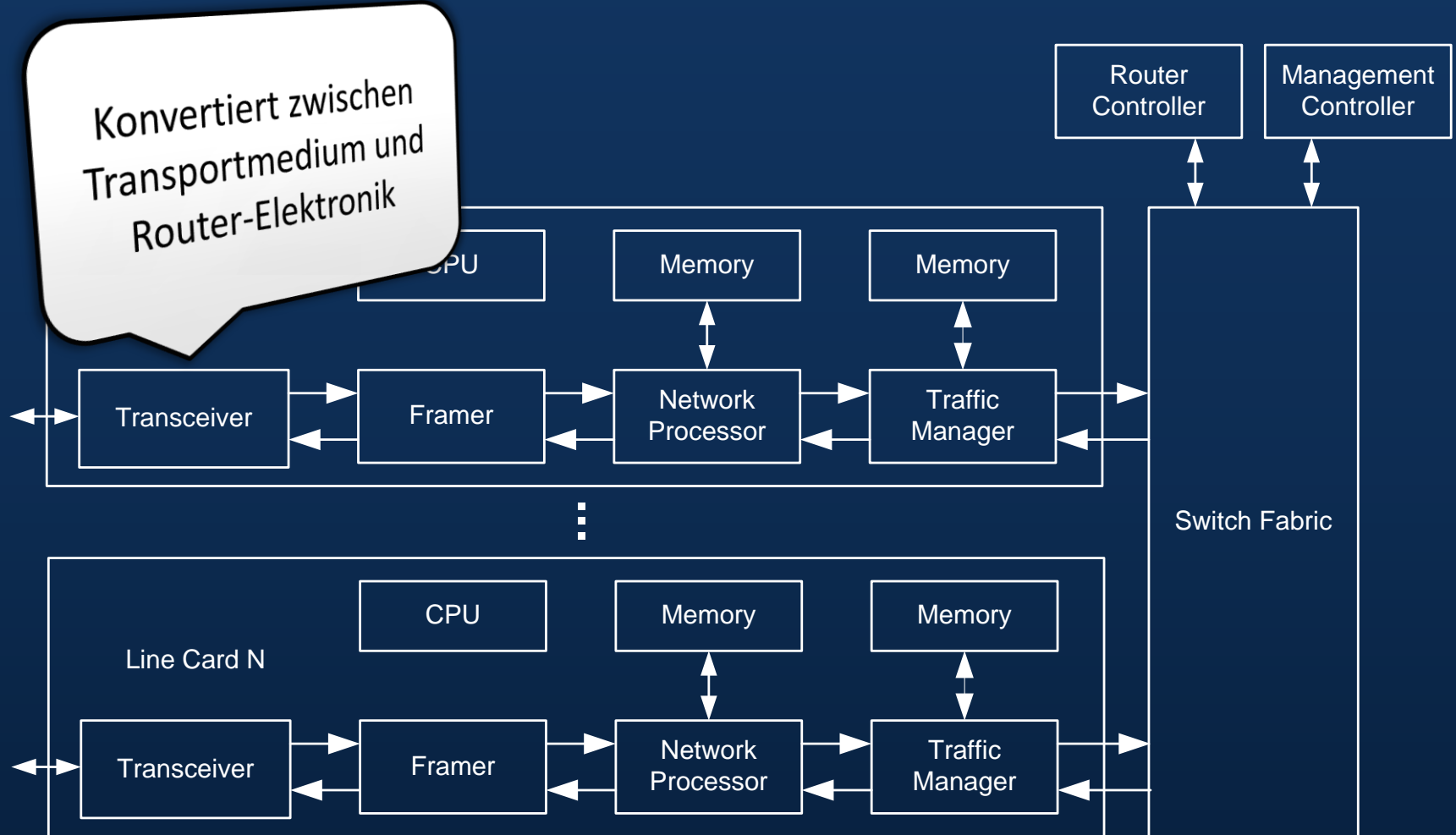
¹ Angenommen wird eine (willkürlich bestimmte) durchschnittliche Paketlänge von 1000 Byte

Quelle: Wikipedia-Eintrag „DE-CIX“

Einleitung / Motivation

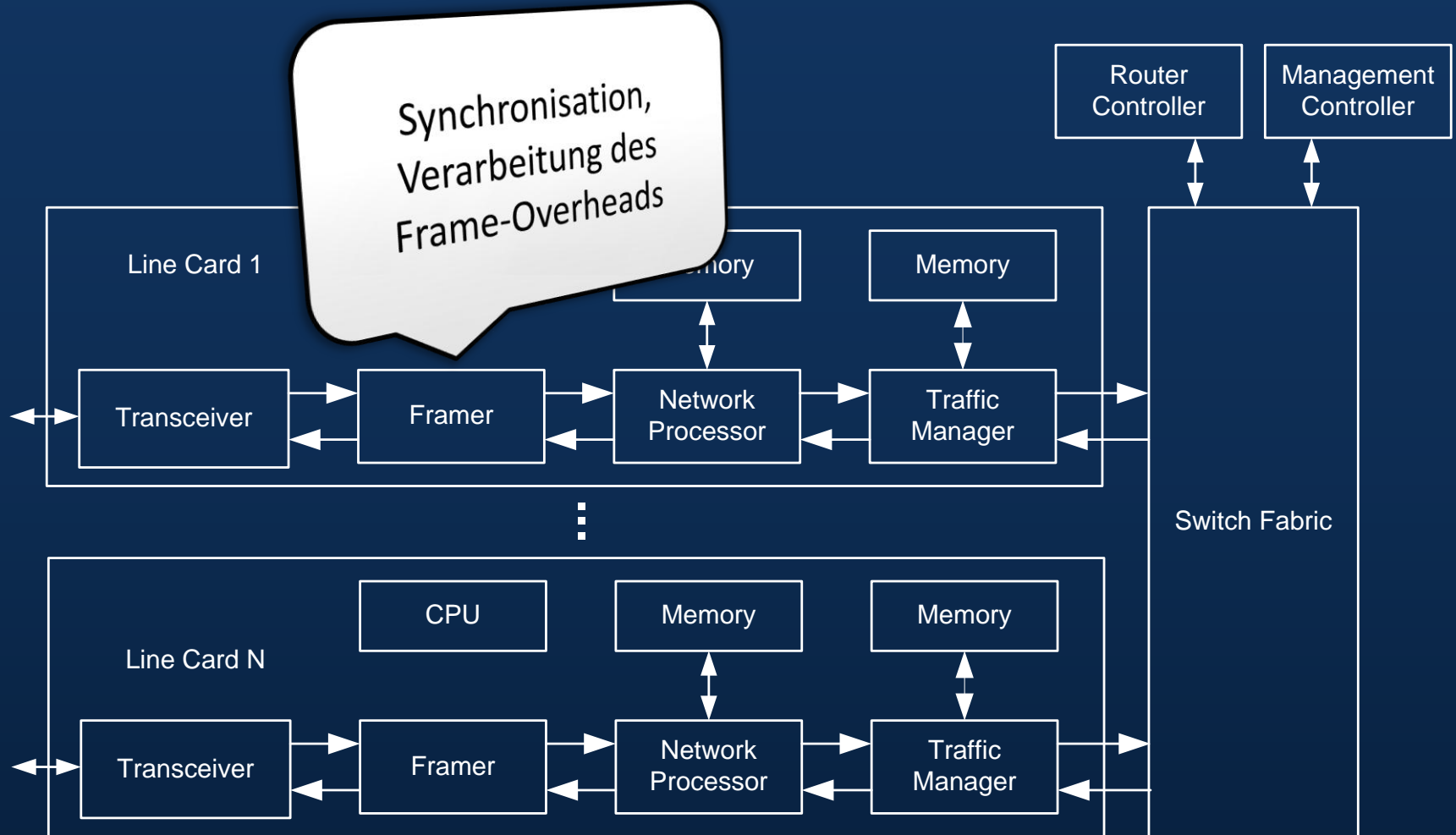
- Auch wenn die meisten IP-Pakete zu logischen TCP-Verbindungen gehören, läuft die Routenermittlung bzw. das Switching für jedes Paket individuell ab
- 62.5 Mio. IP-Pakete erfordern also 62.5 Mio. Lookup-Vorgänge in den Routing-Tabellen – pro Sekunde, d.h. pro Lookup 16 μ s
- Internet-Router haben teilweise >200.000 Einträge in ihren Routing-Tabellen
- Wie erreichen Router die nötige Performance?

Typischer Aufbau eines Routers



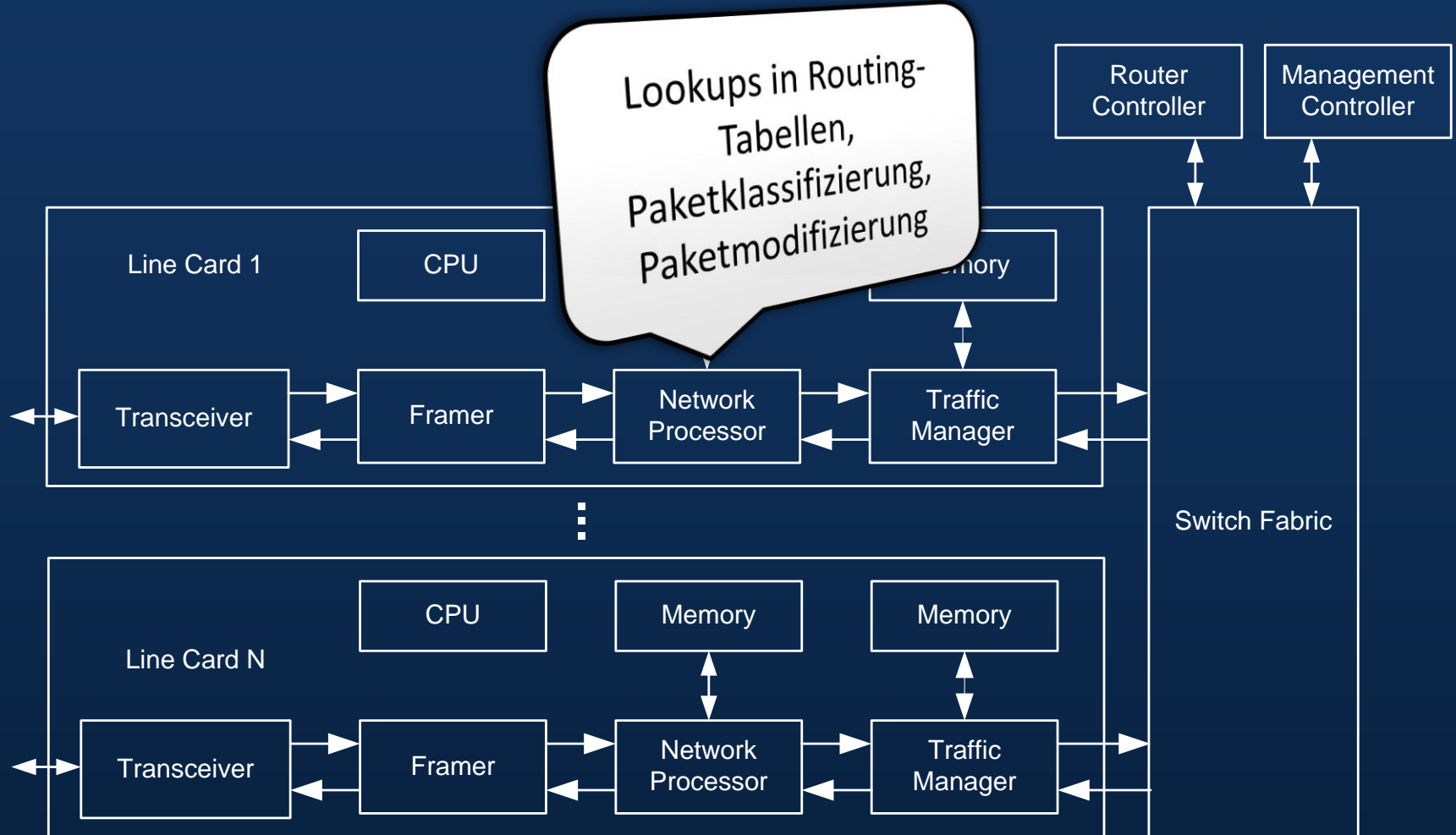
Quelle: H. Jonathan Chao, Bin Liu, High Performance Switches and Routers, Wiley, 2007, Seite 28

Typischer Aufbau eines Routers



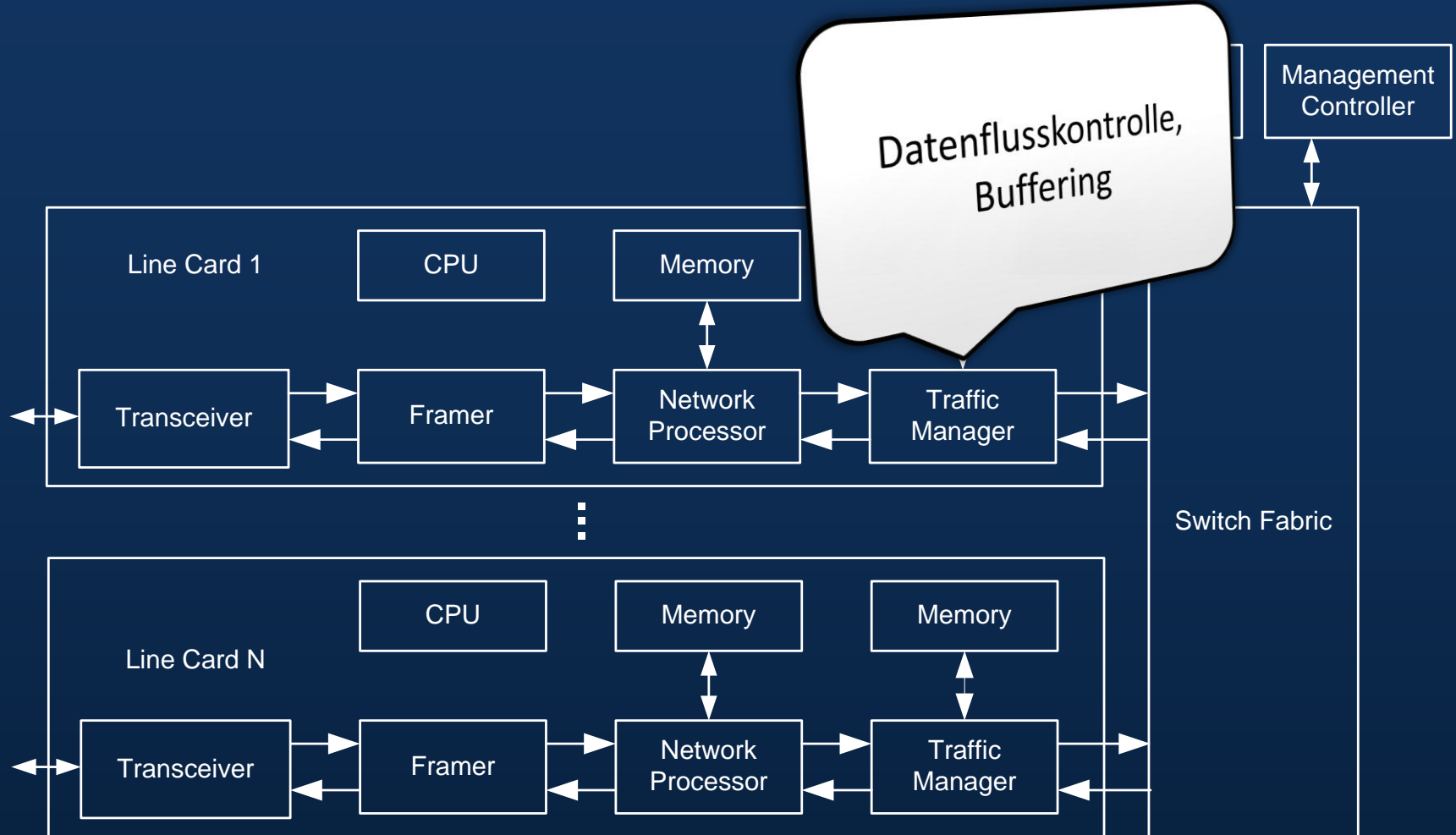
Quelle: H. Jonathan Chao, Bin Liu, High Performance Switches and Routers, Wiley, 2007, Seite 28

Typischer Aufbau eines Routers



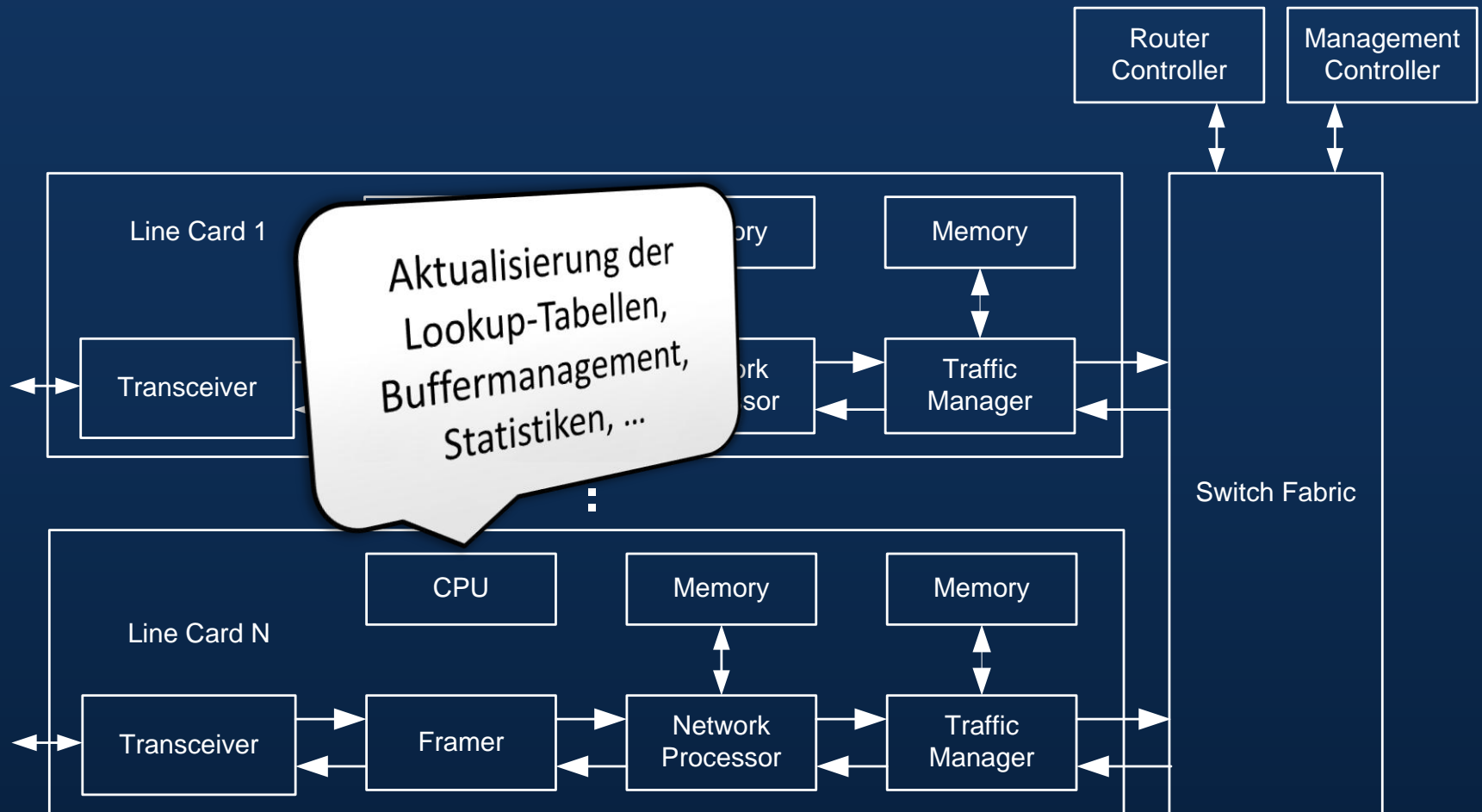
Quelle: H. Jonathan Chao, Bin Liu, High Performance Switches and Routers, Wiley, 2007, Seite 28

Typischer Aufbau eines Routers



Quelle: H. Jonathan Chao, Bin Liu, High Performance Switches and Routers, Wiley, 2007, Seite 28

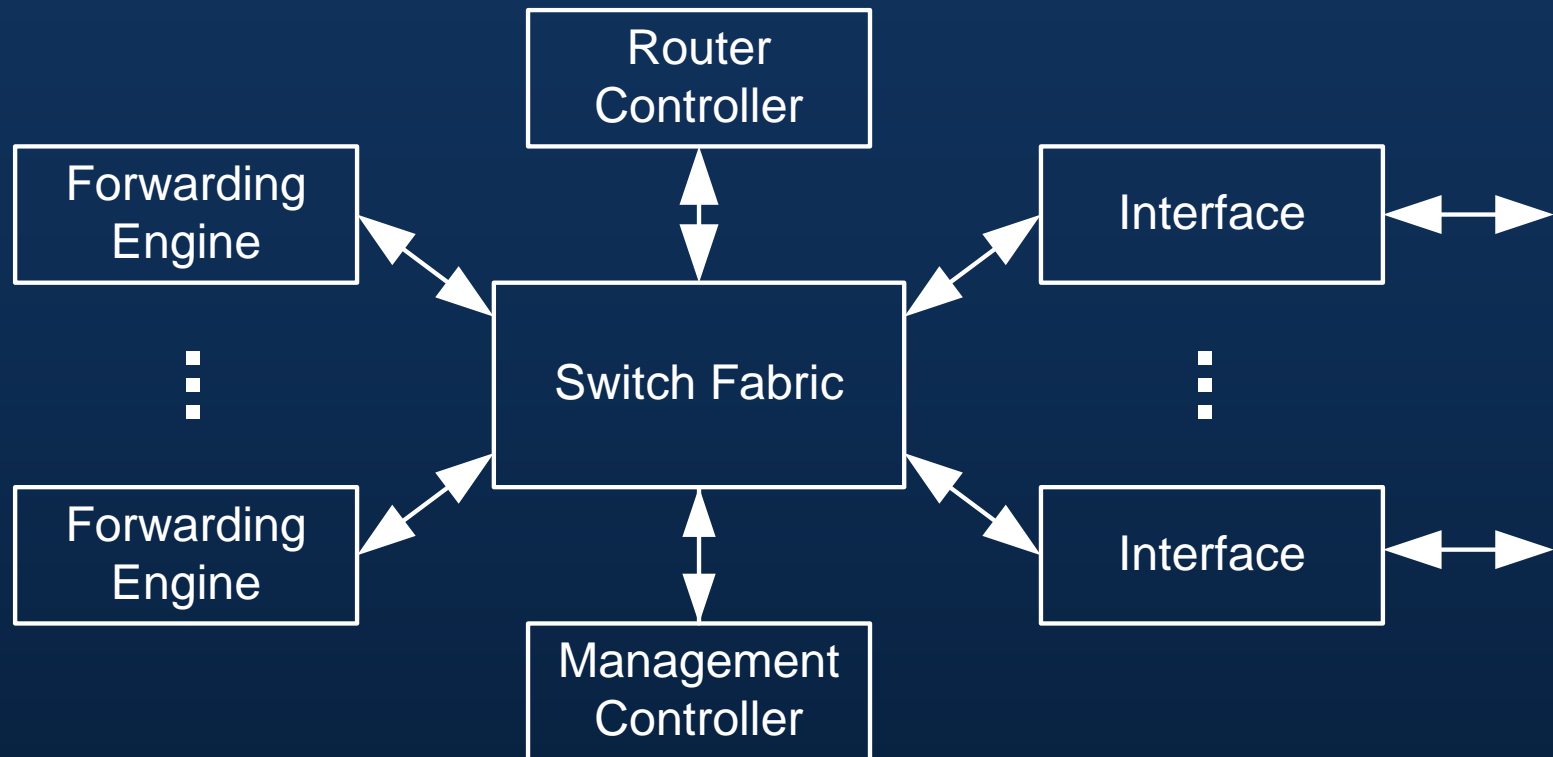
Typischer Aufbau eines Routers



Quelle: H. Jonathan Chao, Bin Liu, High Performance Switches and Routers, Wiley, 2007, Seite 28

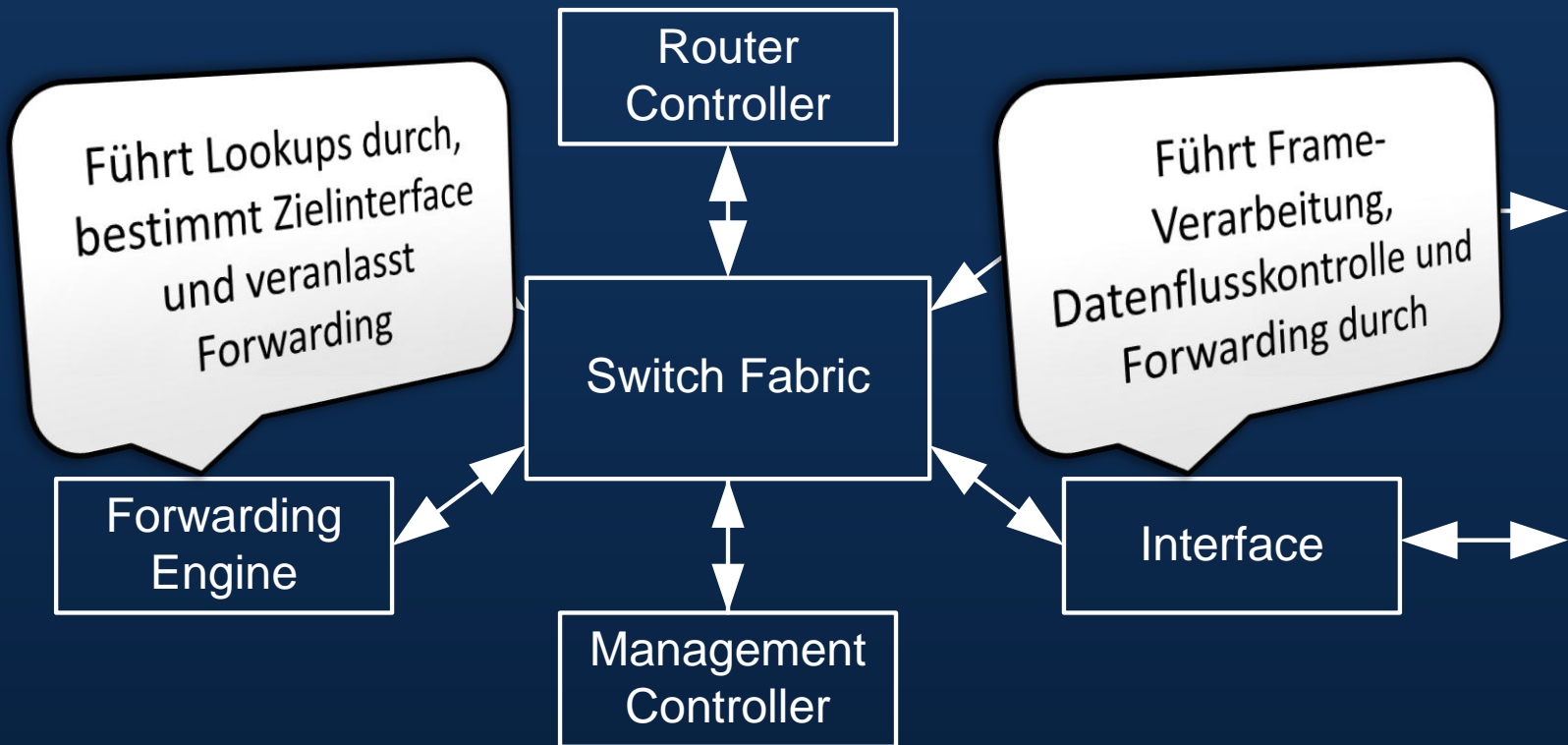
Typischer Aufbau eines Routers

- Günstigere, aber weniger leistungsfähige Alternative:



Typischer Aufbau eines Routers

- Günstigere, aber weniger leistungsfähige Alternative:

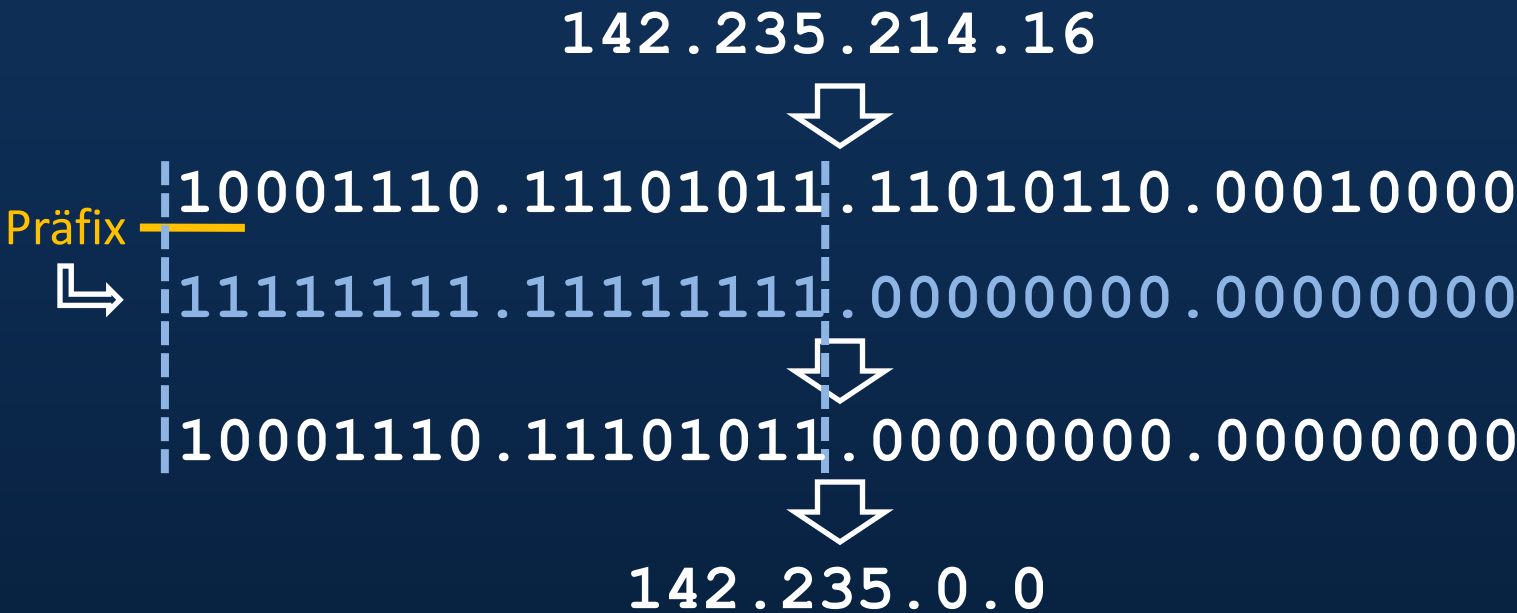


IP-Adress-Lookup

- Die Network Processors eines Routers müssen Entscheidungen fällen:
 - Welches ist der nächste Router für ein Paket?
 - An welches Interface muss ein Paket gesendet werden?
- Diese Informationen werden durch Routing-Protokolle (z.B. BGP) in Routing-Tabellen eingetragen
- Primärschlüssel ist dabei die Zieladresse des Pakets; genauer: das Network-Prefix

IP-Adress-Lookup – Classful Addressing vs. CIDR

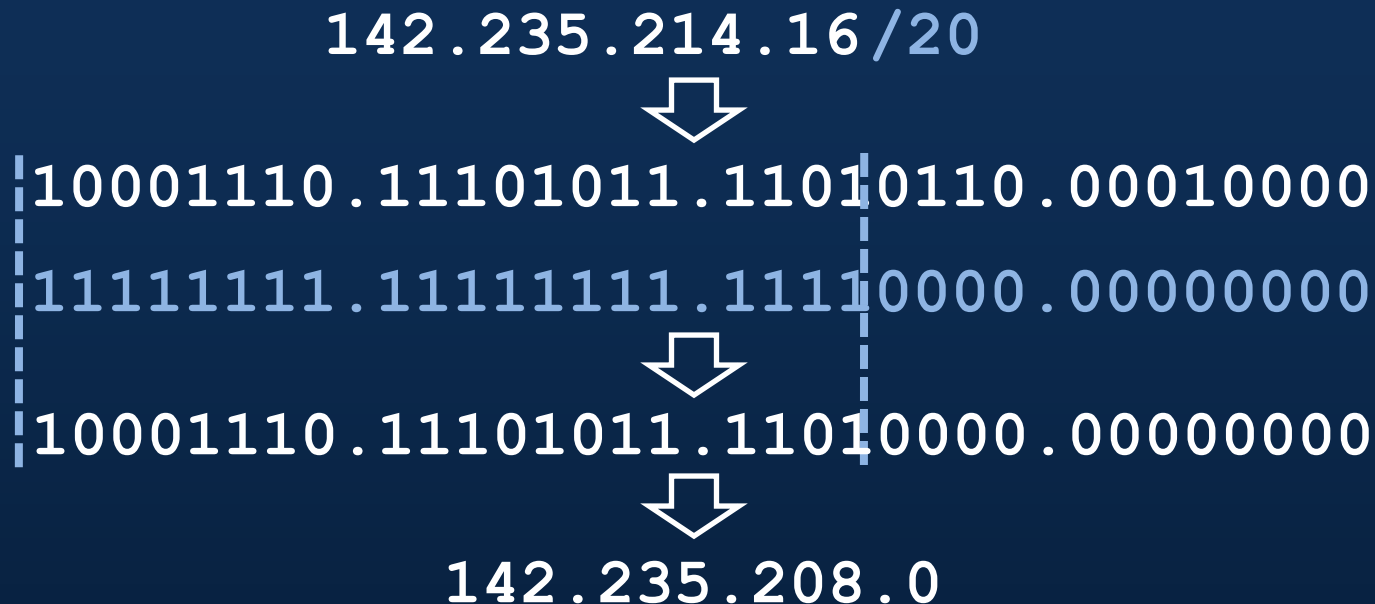
Klasse	Präfix	Adressbereich	Netzmaske
A	0...	0.0.0.0 – 127.255.255.254	255.0.0.0
B	10...	128.0.0.0 – 191.255.255.254	255.255.0.0
C	110...	192.0.0.0 – 223.255.255.254	255.255.255.0



- Die Zielnetzwerkadresse ist allein aus der Ziel-IP bestimmbar

IP-Adress-Lookup – Classful Addressing vs. CIDR

- Bei Nutzung von Classless Inter-Domain Routing ist Kenntnis der Netzmaske zwingend erforderlich



IP-Adress-Lookup

- CIDR ist Fluch und Segen zugleich für Router-Entwickler
 - Da die Länge des Network-Prefix sich nicht mehr aus der Ziel-IP bzw. den Netzwerkadressen selbst ableiten lässt, müssen die Netzwerkadressen mit Angabe der Prefix-Länge in den Routing-Tabellen gespeichert werden
 - Beim Lookup müssen Prefixes unterschiedlicher Länge verglichen werden

IP-Adress-Lookup

- CIDR ist Fluch und Segen zugleich für Router-Entwickler
 - + CIDR erlaubt in vielen Fällen die Zusammenlegung mehrerer Regeln für einzelne Netze zu einer einzigen Regel: *Address Aggregation*

142.235.16.0/24

142.235.17.0/24

⋮

142.235.31.0/24



142.235.16.0/20

10001110.11101011.00010000.0

10001110.11101011.00010001.0

⋮

10001110.11101011.00011111.0

Die ersten 20 Bit sind identisch!

10001110.11101011.00010000.0

IP-Adress-Lookup

- Was aber, wenn ein einziges Zielnetz innerhalb eines Blocks anders geroutet werden muss?
 - Für das „Ausnahme-Netz“ wird einfach eine separate Regel erstellt:
⇒ 142.235.16.0/20 und 142.235.21.0/24
 - Jetzt überlappen sich die Regeln
- Ein CIDR-fähiger Lookup-Algorithmus mit Fähigkeit zur Adressaggregation muss also eine Tabelle nach dem längsten übereinstimmenden Prefix für eine Ziel-IP durchsuchen können!

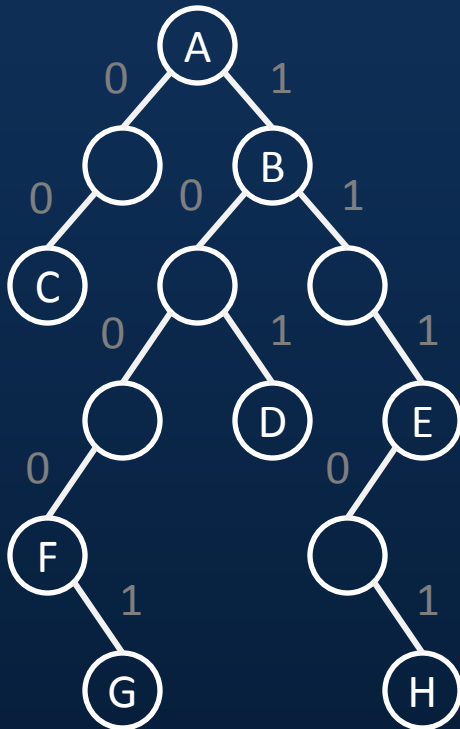
IP-Adress-Lookup - Umsetzung

- Grundsätzlich gibt es zwei gängige Ansätze
 - Trie-basierte Algorithmen
 - Hardware-basierte Umsetzungen¹

¹ Hardware-basiert meint hierbei, dass der Ansatz entweder auf Umsetzung in Hardware optimiert ist oder spezielle Hardware-Komponenten erfordert, die über normale Prozessor- und Speichertechnologie hinausgehen. Selbstverständlich können auch Trie-basierte Algorithmen in Hardware implementiert werden.

IP-Adress-Lookup - Umsetzung - Tries

- Ein Trie („Präfixbaum“) ist eine Datenstruktur zur Suche nach Zeichenketten
- Einfachste Umsetzung: Binärer Trie

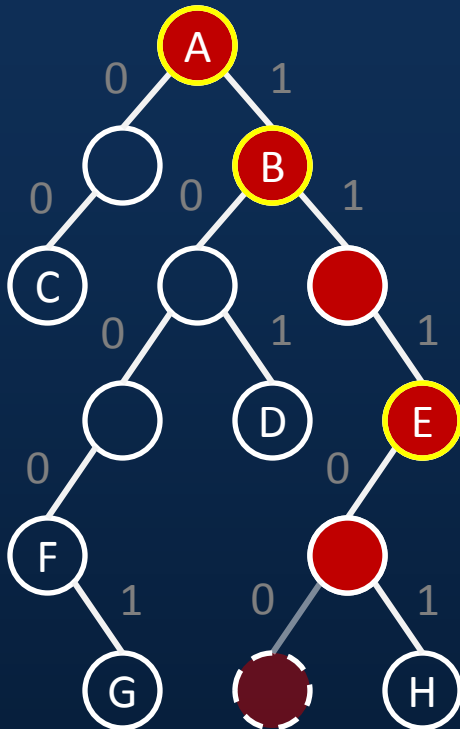


Prefix-Datenbank

A	*
B	1*
C	00*
D	101*
E	111*
F	1000*
G	10001*
H	11101*

IP-Adress-Lookup - Umsetzung - Tries

- Suche im binären Trie
 - Beispiel: IP-Adresse 228.* ist gegeben (11100100.*)



Prefix-Datenbank

A	*
B	1*
C	00*
D	101*
E	111*
F	1000*
G	10001*
H	11101*

IP-Adress-Lookup - Umsetzung - Tries

- Vorteile / Nachteile Binärer Trie
 - + Einfach zu implementieren
 - Schlechte Performance
 - Bei IPv4 bis zu 32 Speicherzugriffe erforderlich
 - Um ein Prefix hinzuzufügen müssen bis zu 32 Nodes erstellt werden

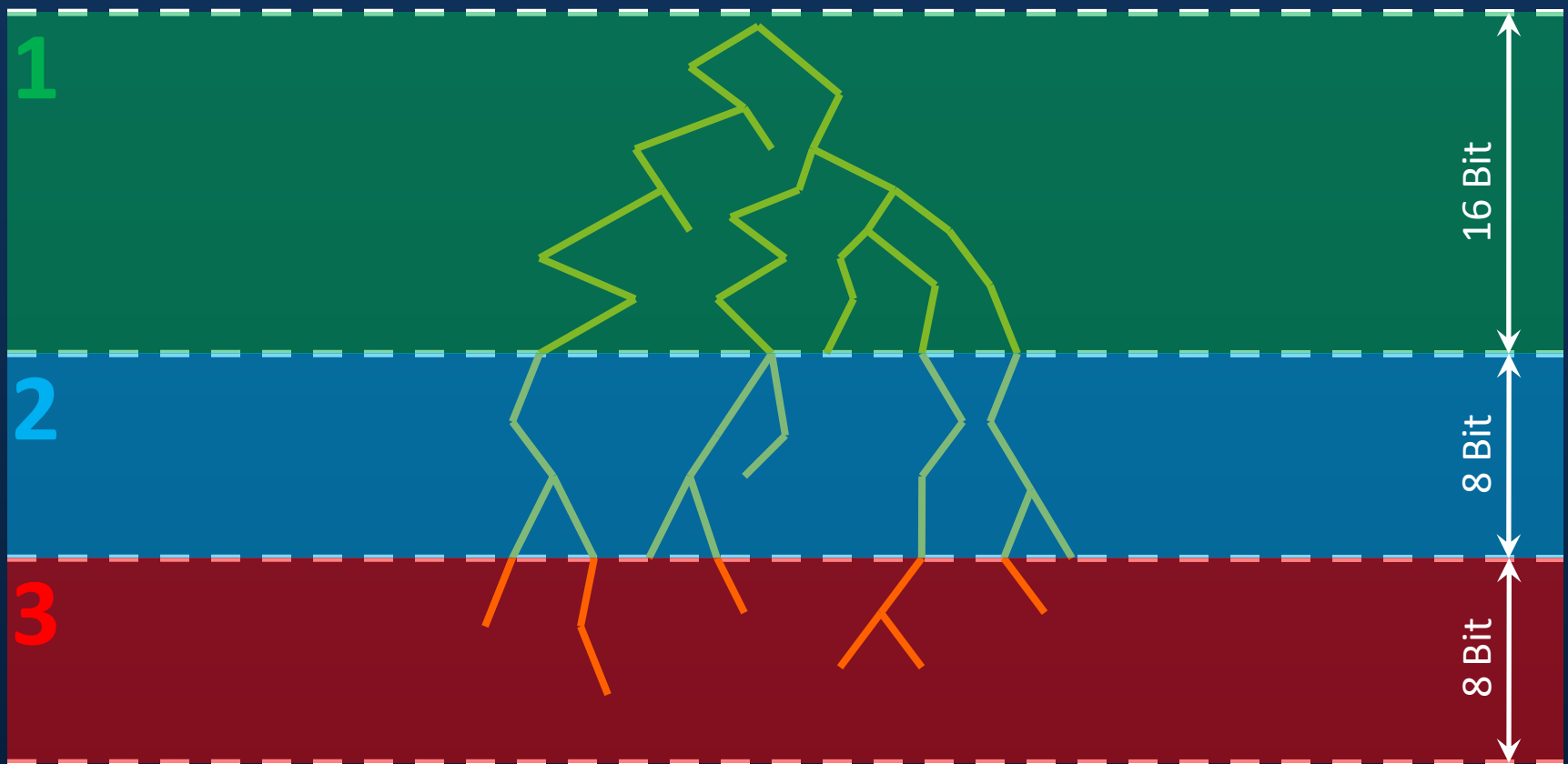
IP-Adress-Lookup - Umsetzung - Tries

- Effizienter ist der Lulea-Algorithmus
 - Entworfen 1997 in Schweden von Mikael Degermark
 - Versucht in erster Linie, hohe Geschwindigkeit durch minimale Datenstrukturgröße zu erreichen
 - Datenstruktur passt auch bei großen Routing-Tabellen vollständig in die lokalen Caches üblicher Prozessoren
 - Leider patentiert :-(

IP-Adress-Lookup - Umsetzung - Tries

- Zunächst: Aufteilung eines 32-Ebenen-Baums in 3 Stufen

142.235.214.16 10001110.11101011.11010110.00010000



IP-Adress-Lookup - Umsetzung - Tries

- Datenstrukturen der ersten Stufe
 - Ein Bit-Vektor bestehend aus $2^{16}=65536$ Bits
 - 1 für jedes 16-Bit-Prefix, für welches Routing-Informationen vorhanden sind (kann auch Teil eines längeren Prefix sein)
 - 0 sonst
 - Ein Array aus 16-Bit-Worten für jede 1 im Bit-Vektor
 - Jedes Datum enthält entweder einen Index in die zweite Stufe oder direkt zu den Routing-Informationen
 - Ein Array von „Basis-Indizes“, einen für jede Subsequenz von 64 Bit im Bit-Vektor (insg. 1024)
 - Zeigt auf das erste zu einer Subsequenz zugehörige Datum

IP-Adress-Lookup - Umsetzung - Tries

- Datenstrukturen der ersten Stufe
 - Ein Array von 16-Bit-„Codeworten“, eines für jede Subsequenz von 16 Bit im Bit-Vektor (insg. 4096)
 - Jedes Codewort besteht aus einem 10-Bit-Wert und einem 6-Bit-Offset
 - Die Summe von Offset plus zugehörigem Basis-Index ergibt einen Pointer zum ersten Datum mit Routing-Informationen in der jeweiligen 16-Bit-Subsequenz
 - Der Wert ist ein Index in die sogenannte „Mactable“ und liefert daraus einen Mactable-Wert. Diese Tabelle enthält die Zahl an Datenworten, die (ausgehend von Basis-Index + Offset) übersprungen werden müssen, um ein bestimmtes Wort zu einem bestimmten Prefix zu finden

IP-Adress-Lookup - Umsetzung - Tries

- Berechnung des Datums-Index für eine gegebene Adresse X
 1. Wert A = Basis-Index an der Position des Basis-Index-Array, die durch die ersten 10 Bits von X gegeben ist
 2. Wert B = Offset aus dem Codewort-Array an der Position, die durch die ersten 12 Bits von X gegeben ist
 3. Wert C = Eintrag der Maptable an der Position gegeben durch den 10-Bit-Wert aus dem Codewort-Array
 4. $\text{Datums-Index} = A + B + C$

IP-Adress-Lookup - Umsetzung - Tries

- Die 2. und 3. Stufe sind auf dieselbe Weise organisiert
 - Der gesamte 8-Bit-Bereich ist in mehrere Chunks unterteilt
 - Wenn ausreichend wenig Einträge in einem Chunk untergebracht werden müssen, werden sie als Liste abgelegt, die sequenziell durchsucht wird
 - Enthält ein Chunk so viele Einträge, dass sich der Overhead beim Lookup lohnt, wird ein Index-Verfahren wie bei der ersten Stufe eingesetzt

IP-Adress-Lookup - Umsetzung - Tries

- Vorteile / Nachteile Lulea-Algorithmus
 - + Gute Performance beim Lookup (wenig Speicherzugriffe, Datenstruktur passt in Prozessor-Cache)
 - + Skalierbar bis zu sehr vielen Einträgen
 - Schlechte Performance beim Aktualisieren
 - In den meisten Fällen ist es schneller, die komplette Datenstruktur neu zu erstellen, statt Aktualisierungen vorzunehmen

IP-Adress-Lookup - Umsetzung - Tries

- Es geht noch besser...
- Tree Bitmaps kombinieren die Vorteile von Lulea mit schnelleren Updates
 - Vorgeschlagen 2004 von W. Eatherton
 - Basiert wie Lulea auf einer möglichst dicht gepackten Datenstruktur, die jedoch leichteres Aktualisieren erlaubt
 - Heutzutage einer der meistgenutzten Algorithmen im Bereich der High-Performance-Router
 - leider verdammt komplex ;-)

IP-Adress-Lookup - Umsetzung - Hardware

- Viele der auf Tries basierenden Verfahren lassen sich teilweise oder komplett in Hardware implementieren
- Darüber hinaus gibt es aber auch Algorithmen, die speziell auf eine möglichst günstige und performante Umsetzung in Hardware optimiert sind
- Zu guter Letzt gibt es spezielle Speichertechnologien, die sich für den IP-Lookup besonders eignen

IP-Adress-Lookup - Umsetzung - Hardware

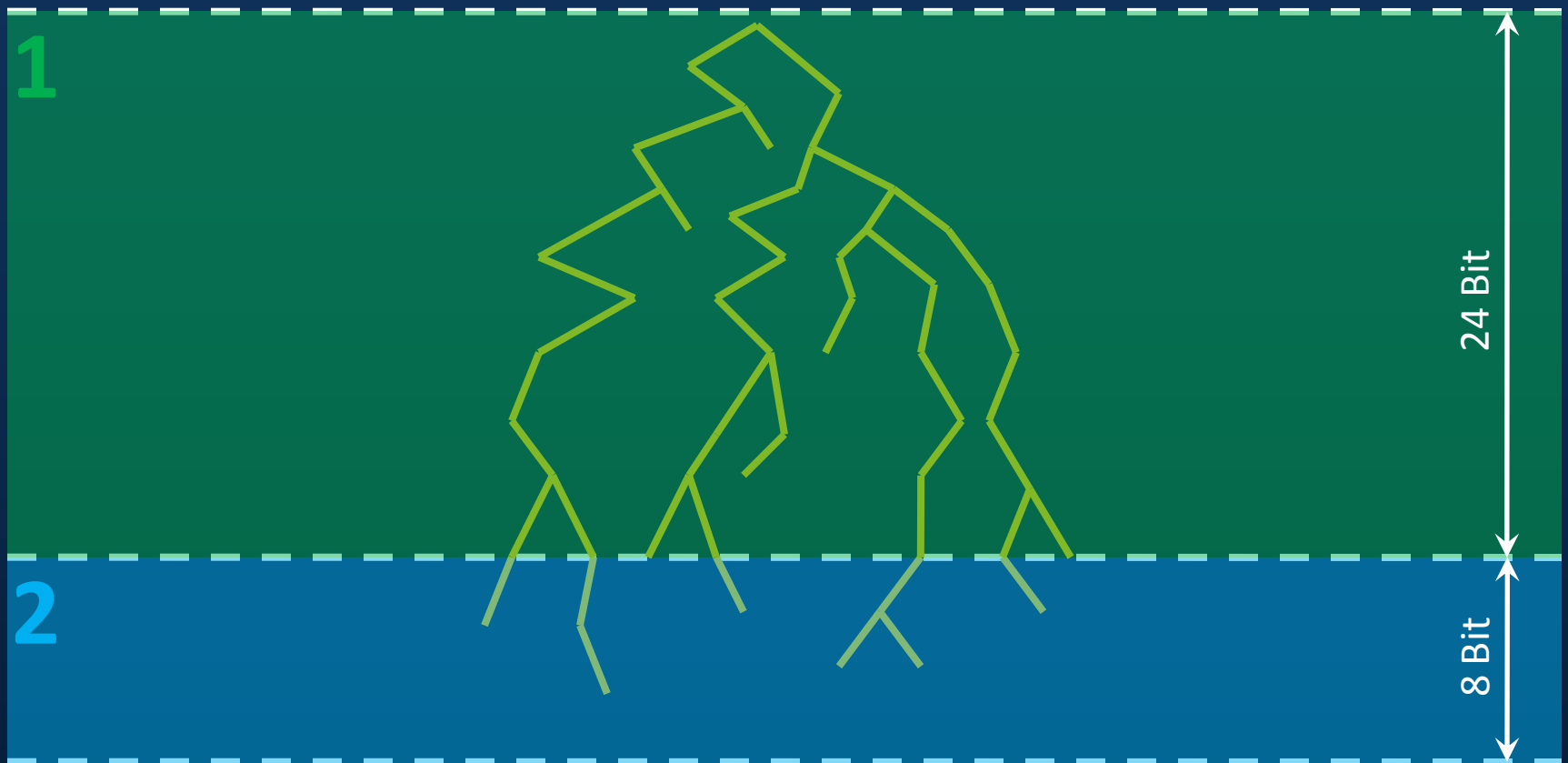
- Das DIR-24-8-BASIC-Schema...
 - ...benötigt 33 Mbyte normalen SDRAMs für den Index
 - ...kann pro Speicherzugriff einen Lookup durchführen
 - ...ist sehr simpel aufgebaut, daher gut und einfach in Hardware zu implementieren

IP-Adress-Lookup - Umsetzung - Hardware

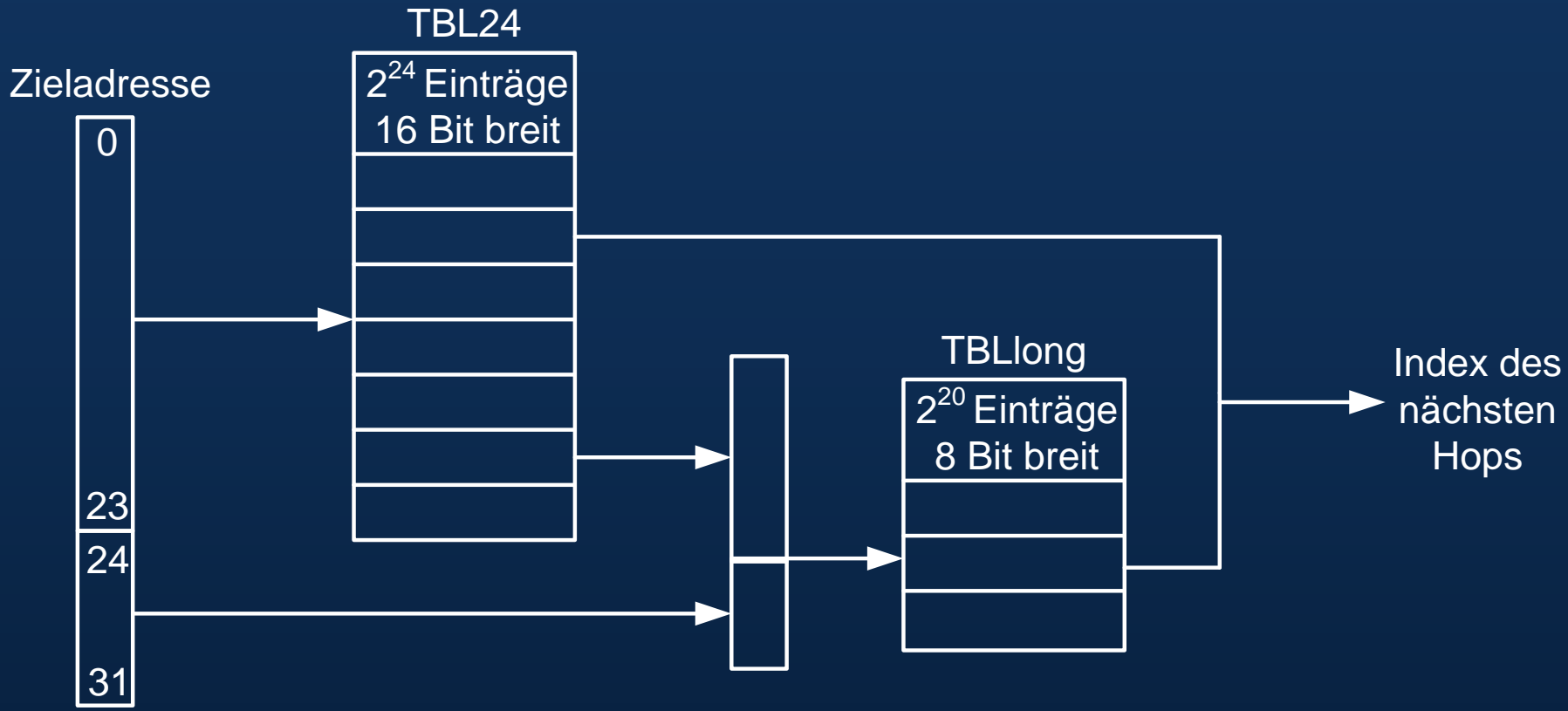
- Aufteilung des Baumes in 2 Stufen

142.235.214.16

10001110.11101011.11010110.00010000



IP-Adress-Lookup - Umsetzung - Hardware



IP-Adress-Lookup - Umsetzung - Hardware

- Ein spezielles Speichermodul: TCAM
 - TCAM = „Ternary Content Adressable Memory“
 - Content Adressable Memory allgemein ist ein Speichertyp, in dem Inhalte nicht über aufsteigende numerische Adressen, sondern ebenfalls über Inhalt adressiert wird
 - Aufzufindender Inhalt: Nächster Hop
 - Adressierender Inhalt: Ziel-IP-Adresse
 - Die Crux: CAM findet nur exakte Übereinstimmungen, gesucht ist aber die Übereinstimmung eines bestimmten Teils: des Network-Prefixes

IP-Adress-Lookup - Umsetzung - Hardware

- Ein spezielles Speichermodul: TCAM
 - Aber „Ternary CAM“ beherrscht dies!
 - Gespeichert wird neben der zu suchenden Netzwerkadresse auch die Netzmaske
 $142.235.0.0/16 \Leftrightarrow (142.235.0.0, 255.255.0.0)$
 - Die Maske bestimmt beim Vergleich, welcher Teil verglichen wird
 - Der Vergleich mit allen gespeicherten Einträgen läuft simultan ab, in einem „Taktschritt“ liegt das Ergebnis vor

IP-Adress-Lookup - Umsetzung - Hardware

- Vorteile / Nachteile TCAM
 - + Sehr gute Performance, pro Takt ein Lookup
 - + Je nach TCAM-Größe skalierbar bis zu sehr vielen Einträgen
 - + TCAM-Chips sind bereits kommerziell verfügbar
 - + Sehr einfach zu implementieren
 - Hoher Energieverbrauch
 - Hoher Preis



(you better keep ignoring him)



MAIK UNSHELM @sm Medien

Datenflusskontrolle und Puffermanagement

DATENFLUSSKONTROLLE UND PUFFERMANAGEMENT

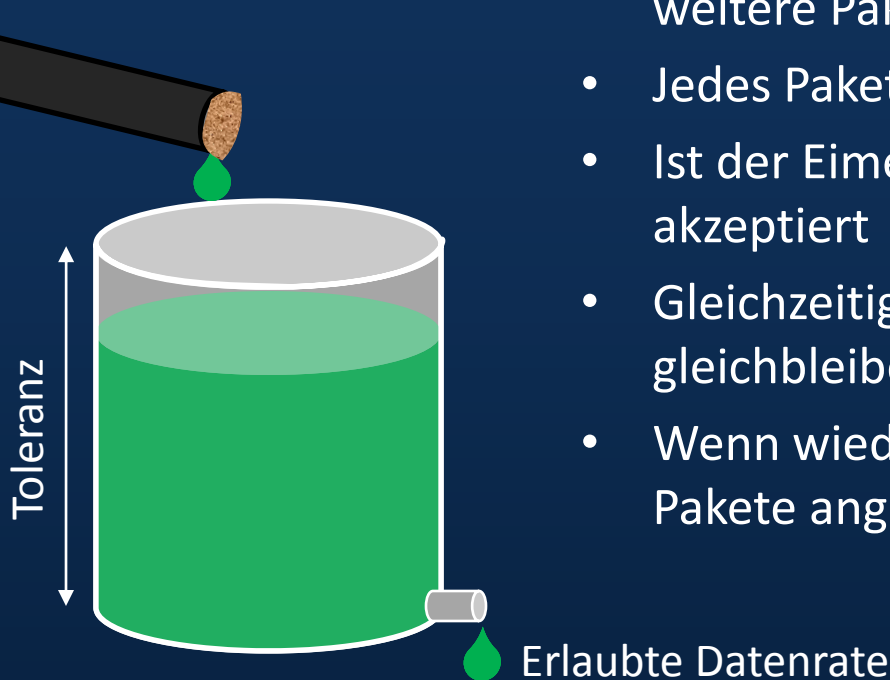
Datenflusskontrolle und Puffermanagement

- Ein wichtiger Bestandteil eines Routers oder Switches ist ein System zur Datenflusskontrolle
 - Sei es zur expliziten Kontrolle, z.B. der Limitierung von Ports auf bestimmte Bandbreiten aufgrund von Policies...
 - ...oder zur impliziten Kontrolle, z.B. weil die Daten von zwei ausgelasteten 100MBit-Input-Ports auf einen einzigen 100MBit-Output-Port geleitet werden müssen
- Für diese Aufgabe kommen unterschiedliche Algorithmen zum Einsatz

Datenflusskontrolle und Puffermanagement

- Ein Beispiel: der Leaky Bucket Algorithm

- So lange, wie freier Platz im Eimer ist, werden weitere Pakete akzeptiert
- Jedes Paket füllt den Eimer
- Ist der Eimer voll, werden keine Pakete mehr akzeptiert
- Gleichzeitig „leckt“ der Eimer mit einer gleichbleibenden Geschwindigkeit
- Wenn wieder Platz im Eimer ist, werden weitere Pakete angenommen



Datenflusskontrolle und Puffermanagement

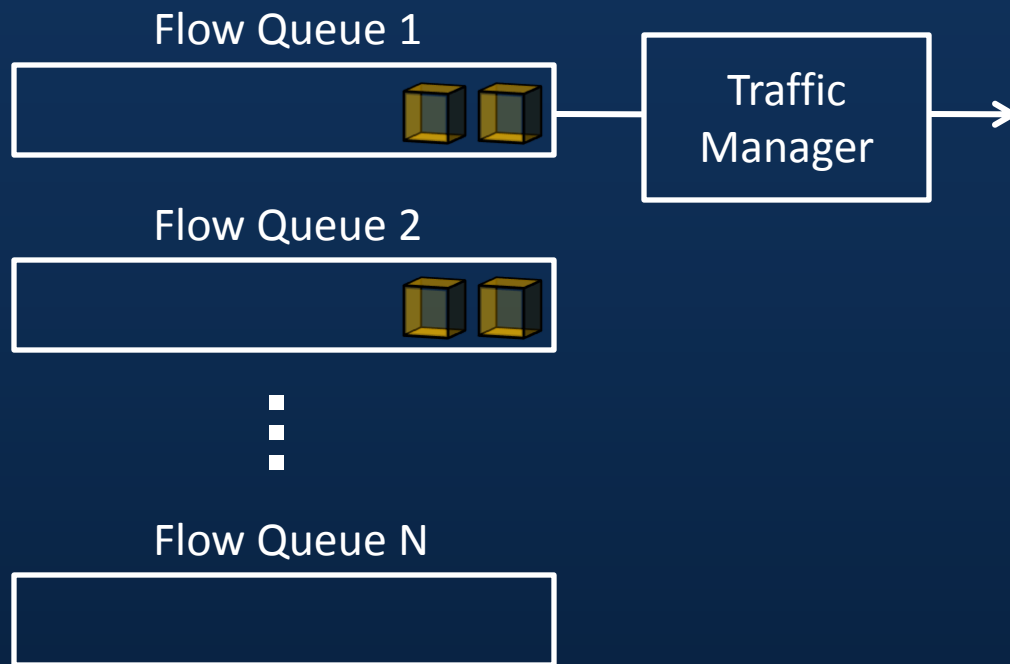
- Algorithmen wie der Leaky Bucket Algorithm werden genutzt, um gezielt Traffic zu begrenzen
- Was aber, wenn mehrere Datenflüsse auf eine Verbindung gemultiplext werden müssen und daher Daten nicht mit voller Geschwindigkeit angenommen werden können?

→ Packet Scheduling

- Auch hier gibt es – wer hätt's gedacht – viele verschiedene in der Praxis eingesetzte Algorithmen

Datenflusskontrolle und Buffer Management

- Einfaches Round-Robin

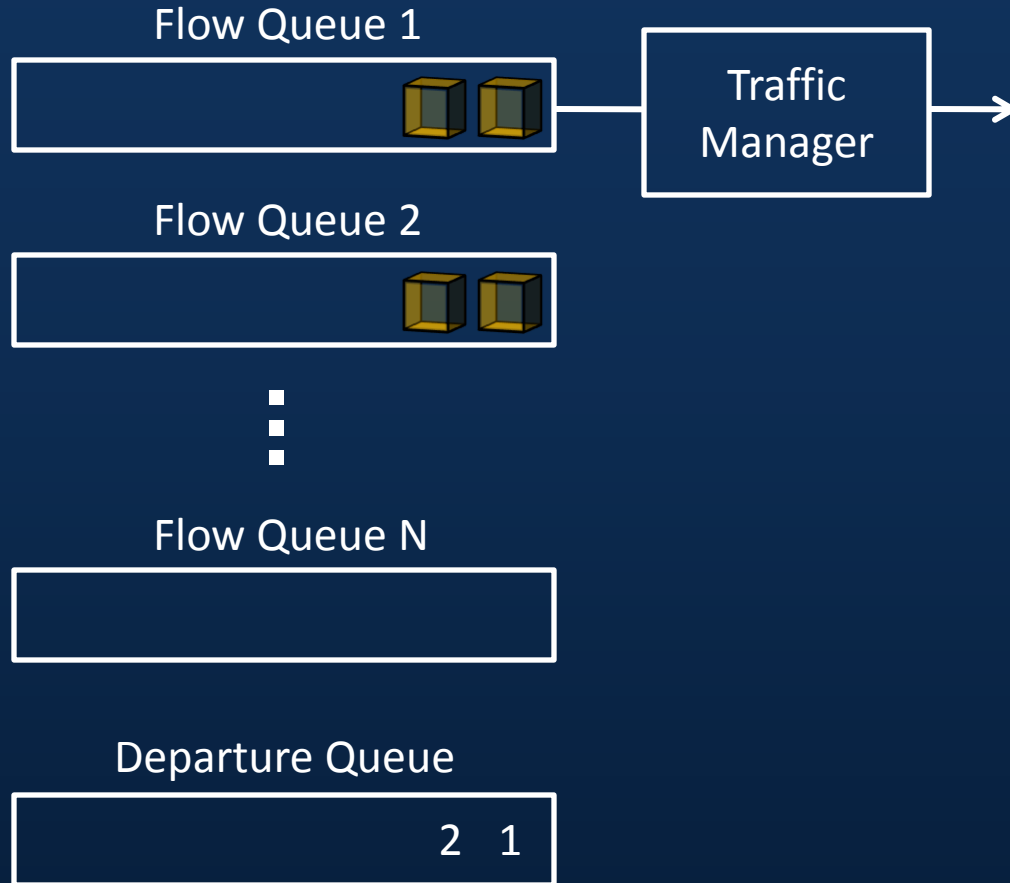


Datenflusskontrolle und Buffer Management

- Der einfache Round-Robin-Algorithmus funktioniert gut, wenn eine gleichmäßige Behandlung mehrerer Verbindungen gewünscht ist, über welche Pakete mit etwa derselben Paketlänge ankommen
- Da das Polling eines Empfangspuffers Zeit benötigt, wird es bei steigender Pufferzahl und Link-Geschwindigkeit jedoch zunehmend problematisch, ständig leere Puffer zu pollen

Datenflusskontrolle und Buffer Management

- Round-Robin mit Departure Queue



Datenflusskontrolle und Puffermanagement

- Round-Robin mit Departure Queue skaliert hervorragend über fast beliebig viele Flow Queues, ohne dabei „Sendezeit“ auf dem Ausgangs-Link mit der Suche nach der nächsten gefüllten Queue zu vergeuden
- Zwei Nachteile bleiben aber noch:
 - Da die Datenrate von der Paketgröße abhängt, ist der Algorithmus nur so lange „fair“, wie die Paketgröße gleich ist
 - Eine gewollt unterschiedliche Gewichtung der Queues ist nicht möglich

Datenflusskontrolle und Puffermanagement

- Weighted Round Robin
 - Arbeitet wie normaler Round-Robin, sendet aber bei jedem „Queue-Besuch“ bis zu X Pakete gleichzeitig
 - X ist für jede Queue separat konfigurierbar, höhere Werte geben einer Queue mehr Output-Bandbreite
 - Beispiel: Queue A mit $X=3$ und Queue B mit $X=7$
 - Einfacher WRR: Reihenfolge „AAABBBBBBB“
 - Bessere Implementierung: „ABBABBABB“

Datenflusskontrolle und Puffermanagement

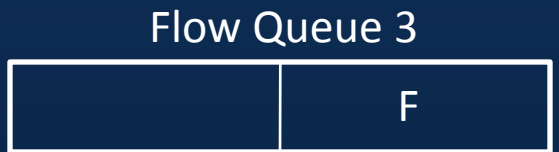
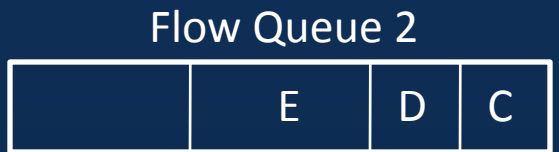
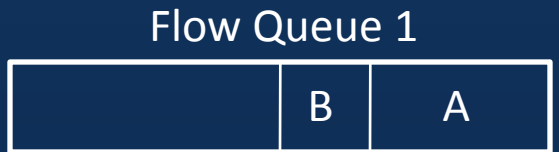
- Deficit Round Robin
 - Arbeitet wie normaler Round-Robin, verwaltet aber pro Queue einen „Deficit Counter“
 - Der Counter wird pro Runde um einen Betrag X erhöht bis zu einem Maximum
 - Ist eine Queue an der Reihe, darf sie so viele Pakete versenden wie sie Bytes im Counter hat. Die gesendeten Bytes werden abgezogen.
 - Kleine Pakete führen so dazu, dass mehrere auf einmal versendet werden bzw. die Queue die noch übrigen Bytes später „nutzen“ kann
 - Ist X pro Queue variabel, ist der Algorithmus „weighted“

Datenflusskontrolle und Puffermanagement

- Weighted Fair Queuing
 - Geht von der Vorstellung aus, dass ein (gewichtetes) Round-Robin möglich ist, bei dem in jeder Runde nur byteweise Daten versendet werden
 - Da dies aber mit Paketen nicht möglich ist, wird dieser Versandvorgang nur „simuliert“; die Pakete werden in der entstehenden Reihenfolge zusammengefügt und dann real versandt

Datenflusskontrolle und Puffermanagement

- Weighted Fair Queuing



Byteweiser Versand, 1. Runde:



Byteweiser Versand, 2. Runde:



Byteweiser Versand, 3. Runde:



Byteweiser Versand, 4. Runde:



Datenflusskontrolle und Puffermanagement

- Weighted Fair Queuing
 - Vervollständigung der Pakete in der Sendewarteschlange „von hinten nach vorne“, d.h. sortiert nach dem Absendezeitpunkt des letzten Teils

Byteweiser Versand, 4. Runde:



Paketweiser Versand



Datenflusskontrolle und Puffermanagement

- Wie die Datenpuffer geleert werden ist nun geklärt, aber was passiert, wenn sie voll sind?
- Der Sender erfährt davon zunächst nichts, er sendet weiterhin so schnell wie möglich!
- Benötigt werden nun Strategien zum Umgang mit den „überzähligen“ Paketen sowie zum Bremsen des Senders

→ Puffermanagement

Datenflusskontrolle und Puffermanagement

- Ein paar Vorbemerkungen zum Thema „Congestion Handling“
 - Sowohl TCP auf OSI-Layer 4 als auch das Ethernet-Protokoll auf OSI-Layer 2 besitzen Mechanismen zur Behandlung von „Verstopfungssituationen“
 - Diese funktionieren jedoch auf grundsätzlich unterschiedliche Weise und ergänzen sich

Datenflusskontrolle und Puffermanagement

- Congestion Handling in TCP
 - TCP erfährt von überlasteten Verbindungen durch Packetloss, d.h. das Ausbleiben von ACK-Paketen
 - Das TCP-Sendefenster erlaubt das Senden einer gewissen Anzahl von Paketen „ins Blaue hinein“
 - Ist das Sendefenster ausgereizt und bleiben ACKs weiterhin aus, stockt die Übertragung und es werden alte Pakete erneut gesendet
 - Darüberhinaus wird bei Paketverlusten die Größe des Sendefensters reduziert, was die Datenrate begrenzt
 - Diese Einschränkung wird bei erfolgreicher Übertragung mit der Zeit Stück für Stück wieder aufgehoben

Datenflusskontrolle und Puffermanagement

- Congestion Handling in TCP



Datenflusskontrolle und Puffermanagement

- Congestion Handling in Ethernet
 - Bei Ethernet existieren spezielle Pause-Frames
 - Empfängt ein Ethernet-Interface ein Pause-Frame, pausiert es das Senden weiterer Daten für die im Frame angegebene Zeit
 - Da Pause-Frames nur im Vollduplexbetrieb möglich sind, wird im Halbduplexbetrieb ein Verfahren namens „Back-Pressure“ genutzt
 - Hierbei simuliert ein Switch Kollisionen, indem er das JAM-Signal sendet
 - Daraufhin tritt die normale Collision-Avoidance-Strategie in Kraft, die dafür sorgt, dass das sendende Interface eine gewisse Zeit wartet, bis es weitere Daten sendet

Datenflusskontrolle und Puffermanagement

- Tail-Drop-Verfahren
 - Einfachstes denkbares System
 - Datenpakete werden einfach ignoriert, sobald ein Empfangspuffer vollgelaufen ist
 - Ignorierte Pakete = Packetloss, woraufhin die Congestion-Avoidance-Mechanismen in TCP die Senderate herunterfahren
 - Im Fall von Ethernet-Switches ist das Ignorieren von Paketen i.d.R. nicht nötig, da durch Pause-Frames oder das Back-Pressure-Verfahren ein kurzzeitiger Stopp der Übertragung veranlasst werden kann

Datenflusskontrolle und Puffermanagement

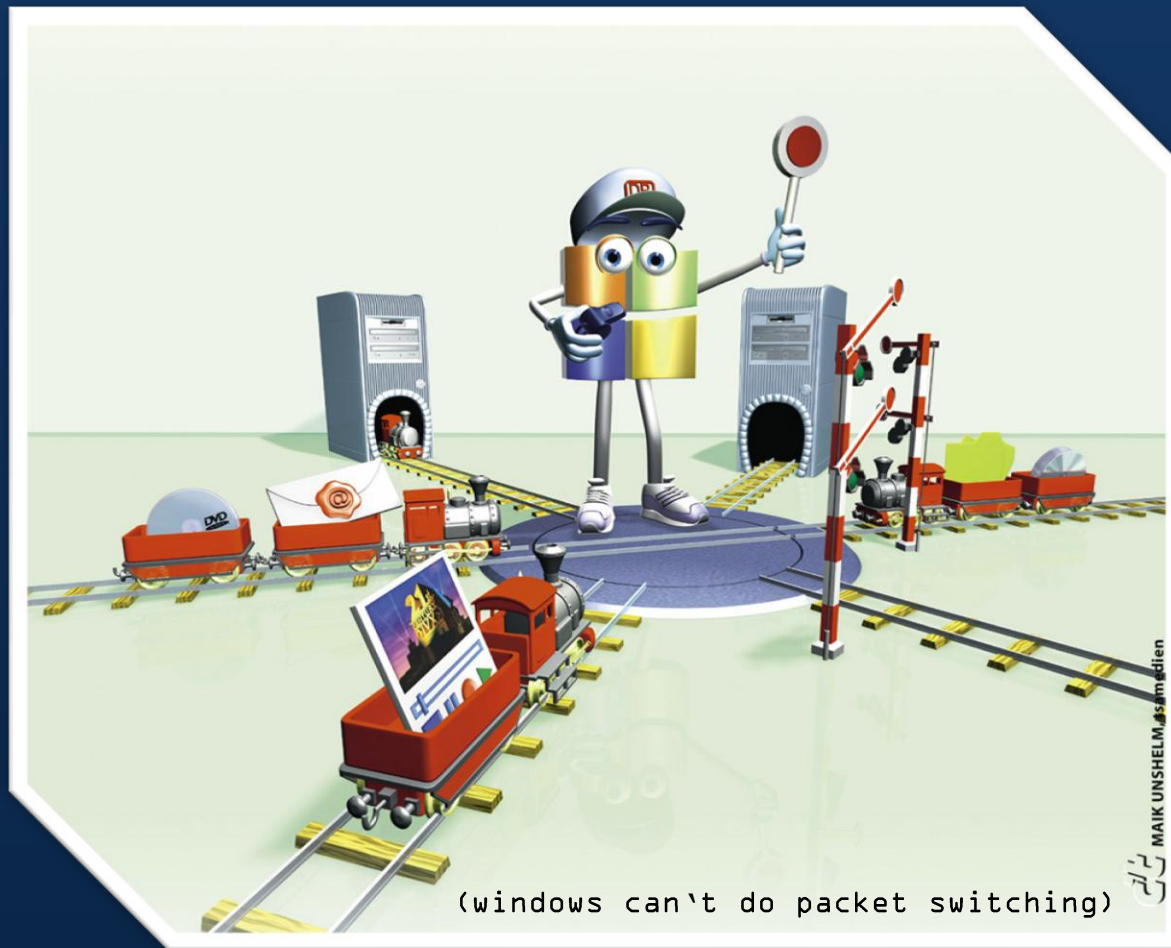
- [Random/Front] Drop on Full
 - Grundsätzlich so einfach wie Tail Drop: wenn die Queue voll ist, werden Pakete gedroppt
 - Random Drop on Full löscht zufällig ein Paket aus der Queue, wenn ein weiteres ankommt
 - Front Drop on Full löscht das vorderste Paket
 - Die Idee: wird ein „älteres“ Paket gelöscht, reagiert TCP schneller auf die Überlastsituation, weil es schneller davon erfährt

Datenflusskontrolle und Puffermanagement

- Random Early Detection
 - Nachteil der beiden genannten Verfahren: Maßnahmen werden erst ergriffen, wenn eine Queue überläuft
 - Ideal wäre es, wenn die Sender die Senderate schon senken würden, bevor eine Überlastsituation entsteht
 - RED löscht Pakete nach dem Zufallsprinzip, wobei die Wahrscheinlichkeit, dass ein Paket gelöscht wird umso höher ist, je weiter der Eingangspuffer vollläuft

Datenflusskontrolle und Puffermanagement

- Random Early Detection
 - Durch Minimal- und Maximalfüllstände wird ein Fenster definiert, innerhalb dessen der Zufall Pakete zum Löschen auswählt
 - Unter dem Minimal-Füllstand wird kein Paket gelöscht
 - Über dem Maximal-Füllstand wird jedes Paket gelöscht
 - Ziel: das Verkehrsaufkommen bei hoher Last zwischen dem Minimal- und Maximalwert zu halten



(windows can't do packet switching)

MAIK UNSHELM, @sm Medien

Packet Switching

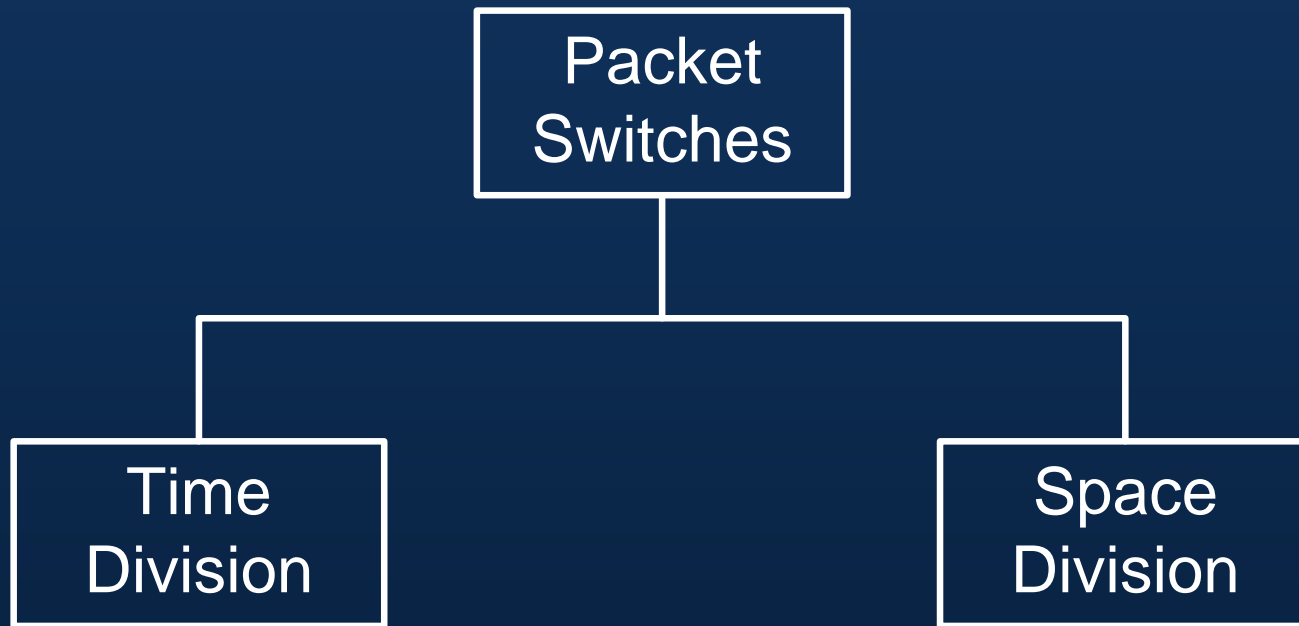
PACKET SWITCHING

Packet Switching

- Switches als separate Netzwerkkomponente arbeiten ähnlich wie die Switch Fabrics, die Teil jedes Routers sind
- Kernaufgabe einer Switch Fabric bzw. eines Switches: Pakete von einem Eingangsport zu einem Ausgangsport leiten
- Die Schwierigkeit besteht in erster Linie in der hohen benötigten Geschwindigkeit

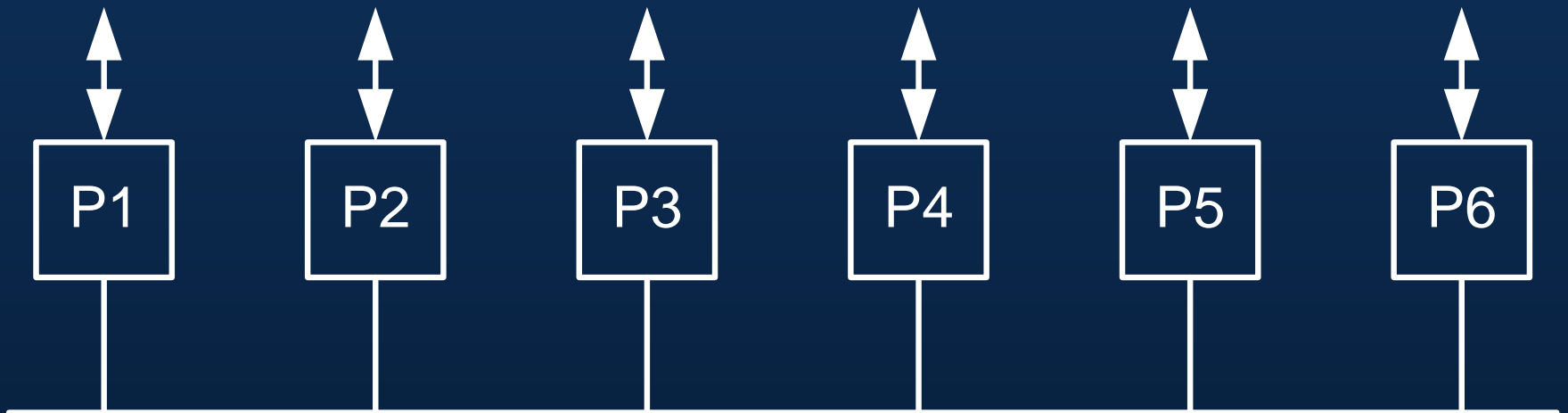
Packet Switching

- Es gibt zwei grundlegende Switching-Techniken



Packet Switching

- Time-Division-Switches
 - Pakete von verschiedenen Quellports werden auf einem gemeinsamen Datenkanal zu den Zielports geleitet



Packet Switching

- Time-Division-Switches
 - Die Klasse der Time-Division-Switches unterteilt sich in Shared-Medium und Shared-Memory
 - Shared-Medium-Switches nutzen einen Bus oder einen Ring als gemeinsamen Datenkanal
 - Jeder Port erhält alle Pakete
 - Ein Filter am Eingang zu jedem Port lässt nur diejenigen Pakete passieren, die an den Port adressiert sind
 - Das geteilte Medium sollte eine Übertragungsrates besitzen, die größer als die summierte Bandbreite aller Ports ist, ansonsten bremst der Switch bei voller Auslastung

Packet Switching

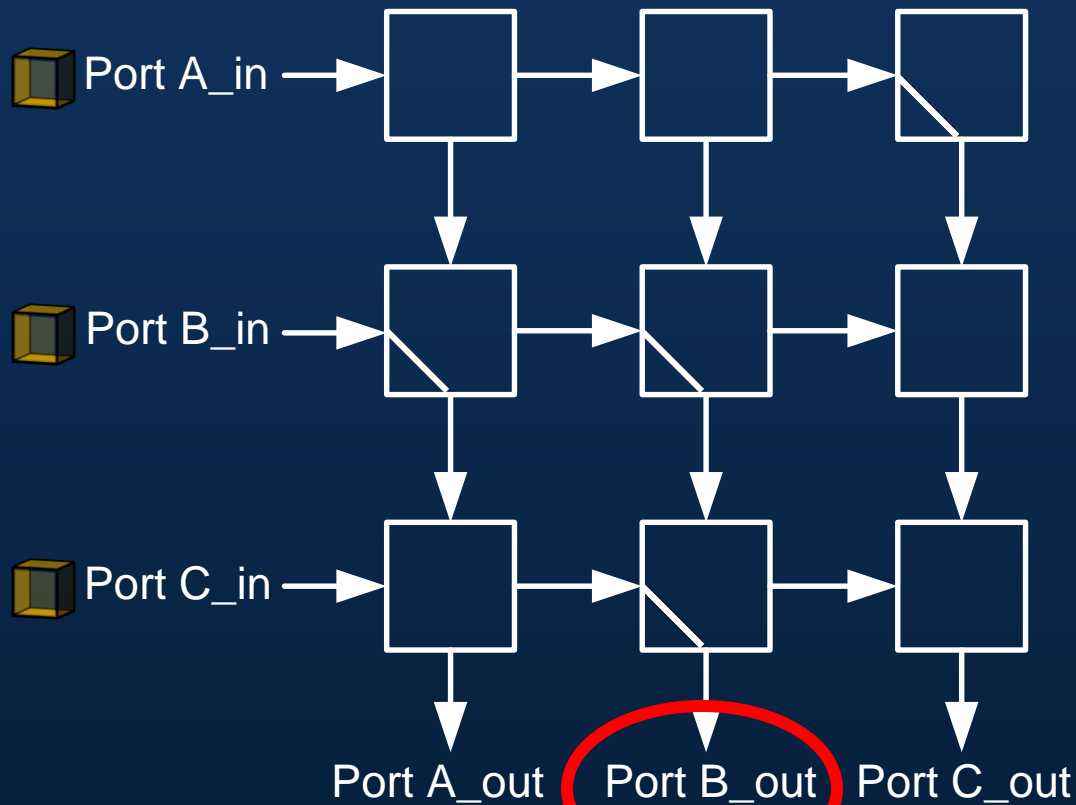
- Time-Division-Switches
 - Die Klasse der Time-Division-Switches unterteilt sich in Shared-Medium und Shared-Memory
 - Shared-Memory-Switches nutzen gemeinsamen Speicher als Datenkanal
 - Alle Ports haben Zugriff auf den gemeinsamen Speicher
 - Pro kleinstem Zeittakt kann ein Port ankommende Pakete in den Speicher schreiben oder für ihn bestimmte Pakete daraus lesen
 - Die Anzahl der Ports ist beschränkt durch die Geschwindigkeit des Speichers; er muss schnell genug sein, um alle Ports mit deren Maximalbandbreite zu versorgen

Packet Switching

- Space-Division-Switches
 - Diese Switch-Klasse zeichnet sich dadurch aus, dass mehrere Pfade von den Input- zu den Output-Ports verfügbar sind, über welche zur selben Zeit Daten übertragen werden können
 - Unterteilt wird hier wiederum in Single-Path und Multi-Path
 - Single-Path-Switches besitzen von jedem Port zu jedem anderen Port nur einen Datenpfad
 - Multi-Path-Switches besitzen mehr als einen Datenpfad zwischen jedem Port-Paar

Packet Switching

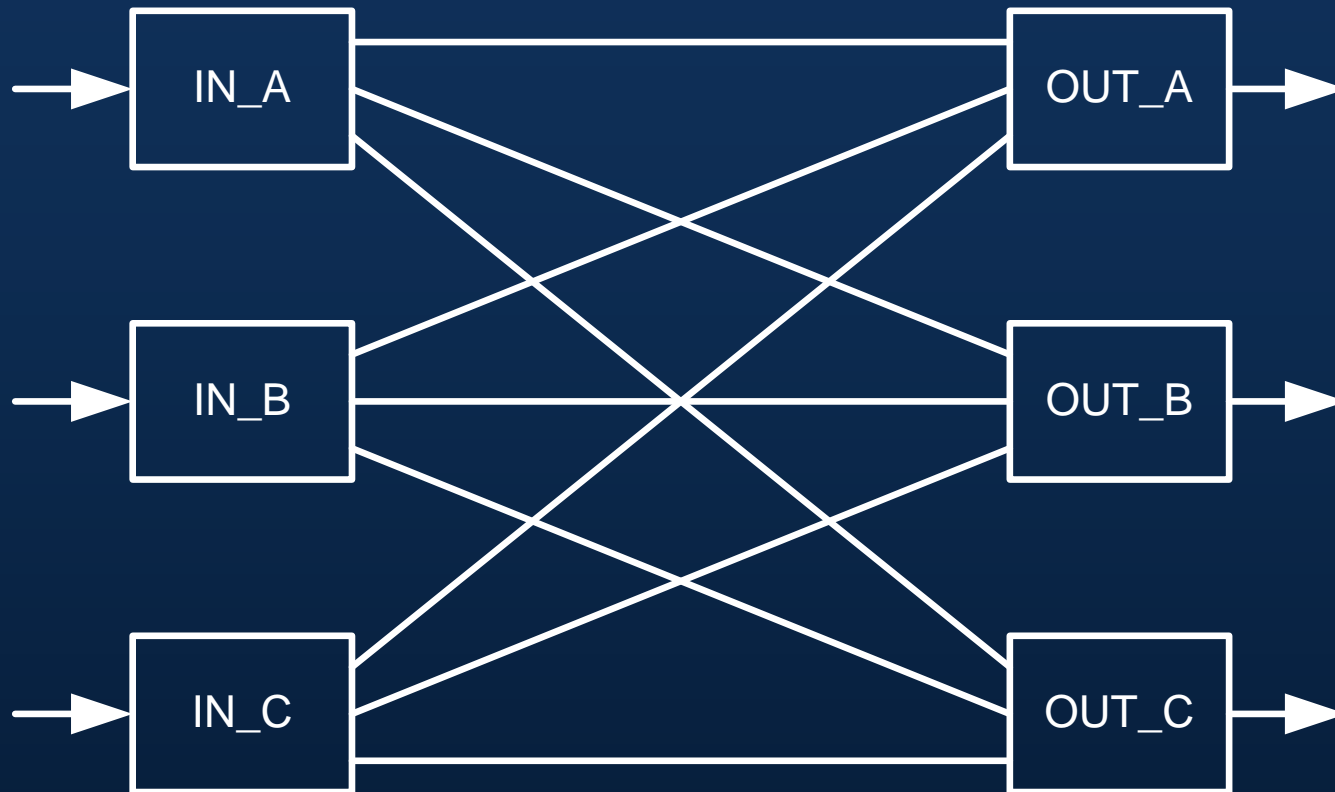
- Space-Division-Switches, Single-Path
 - Crossbar-basierte Switches



Pakete stauen sich!

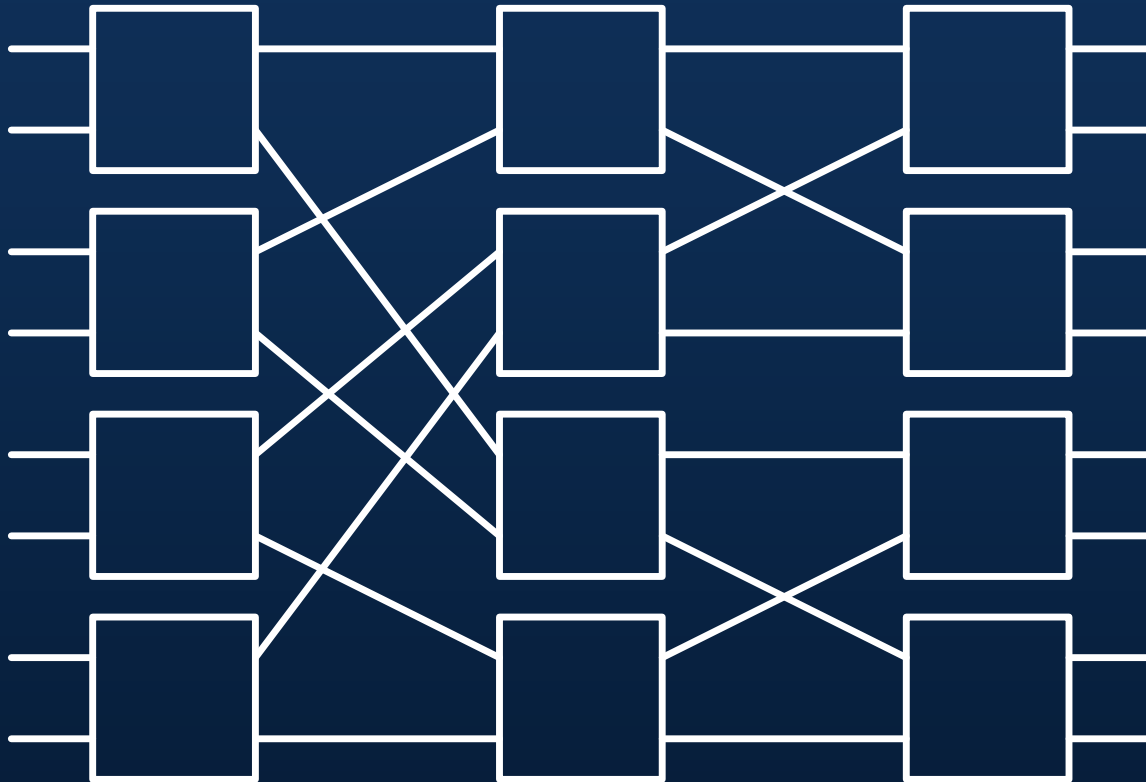
Packet Switching

- Space-Division-Switches, Single-Path
 - Fully Interconnected Switches



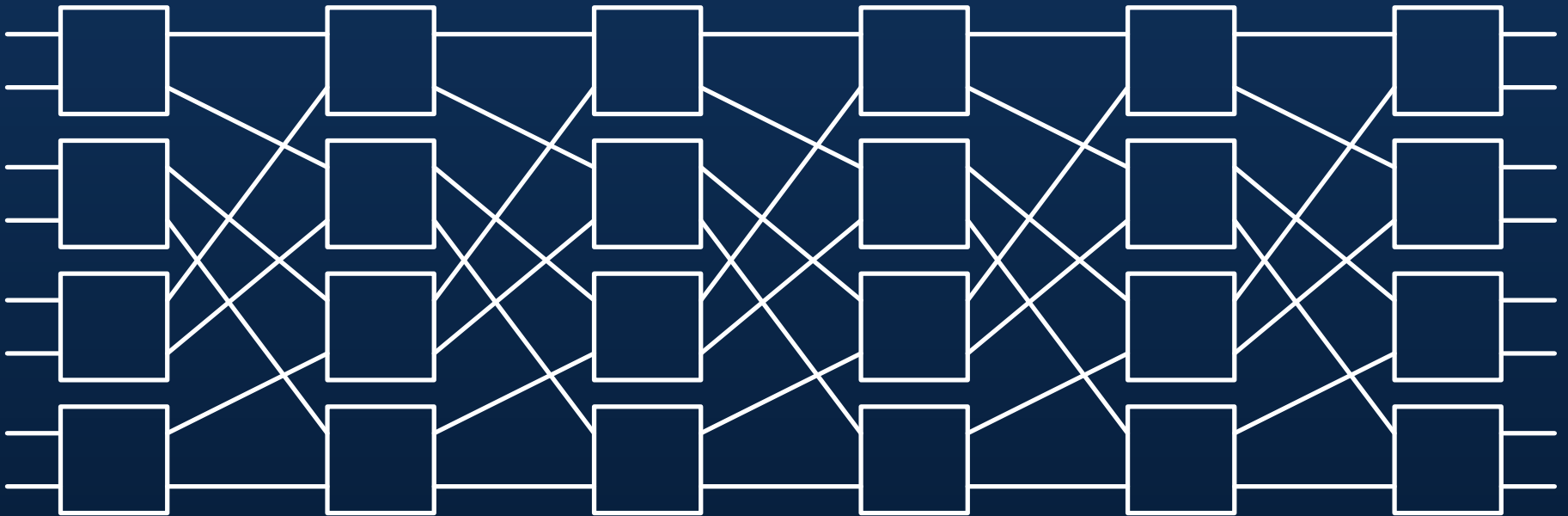
Packet Switching

- Space-Division-Switches, Single-Path
 - Banyan-based Switches
 - Bestehen aus vielen 2x2-Switching-Elementen



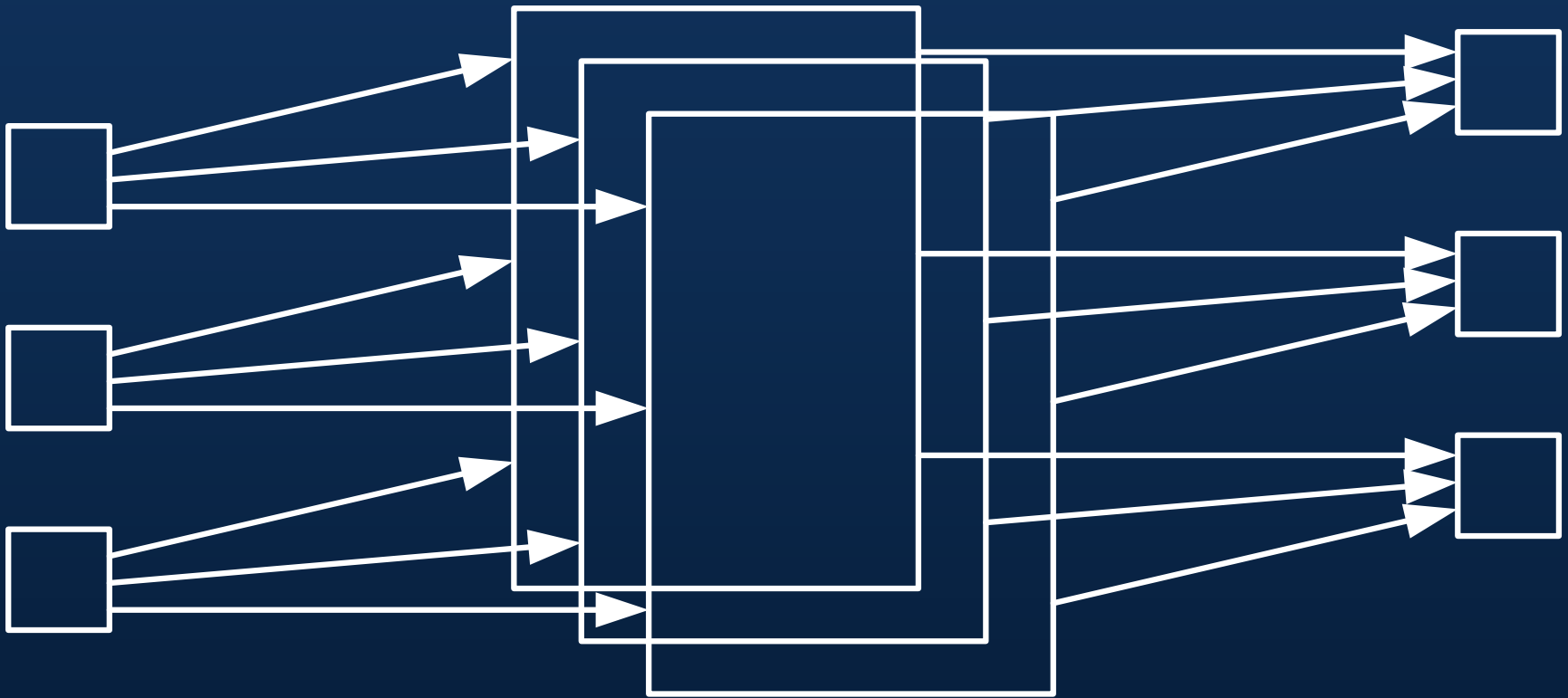
Packet Switching

- Space-Division-Switches, Multi-Path
 - Augmented Banyan Switches
 - Bestehen wie Banyan-Switches aus 2x2-Switching-Elementen
 - Es gibt aber mehrere Pfade von jedem Input- zu Output-Port



Packet Switching

- Space-Division-Switches, Multi-Path
 - Multiplane-Switches



Packet Switching

- Vergleich der Ansätze
 - Time-Division-Switches sind generell einfacher um weitere Ports zu erweitern, da die Switching-Technik in der Regel gemeinsam von allen Ports genutzt wird
 - Das geht jedoch nur so lange gut, wie die Kapazität der Backplane für alle angeschlossenen Ports ausreicht
 - Ab einem gewissen Punkt wird eine Erweiterung sehr teuer, da teure Highspeed-Speicher- bzw. Bustechnik eingesetzt werden muss
 - Nachteil der Time-Division-Verfahren: der Switch besitzt keinerlei eingebaute Redundanz (sofern nicht explizit implementiert)

Packet Switching

- Vergleich der Ansätze
 - Space-Division-Switches erfordern keine Busse oder Speichermodule mit einem Vielfachen der Portgeschwindigkeit, dafür sind sie i.d.R. aufwendiger in der Konstruktion, vor allem bei vielen Ports
 - Bei vielen Space-Division-Techniken steigt die Anzahl interner Verbindungen quadratisch mit der Portanzahl
 - Die Multipath-Varianten besitzen bereits eingebaute Redundanzen, die manche internen Ausfälle kompensieren können

Quellen & Literatur

- Leider gibt es sehr wenig brauchbare Literatur zum Thema – online/kostenlos erst recht nicht
- Empfehlenswertes Buch:
 - High Performance Switches and Routers von H. Jonathan Chao und Bin Liu, Wiley, 2007
 - Geht sehr in die Tiefe und behandelt auf 630 Seiten so ziemlich jeden denkbaren Aspekt des Themas

The End



NAPTIME

Don't fight it any more, the presentation's over!