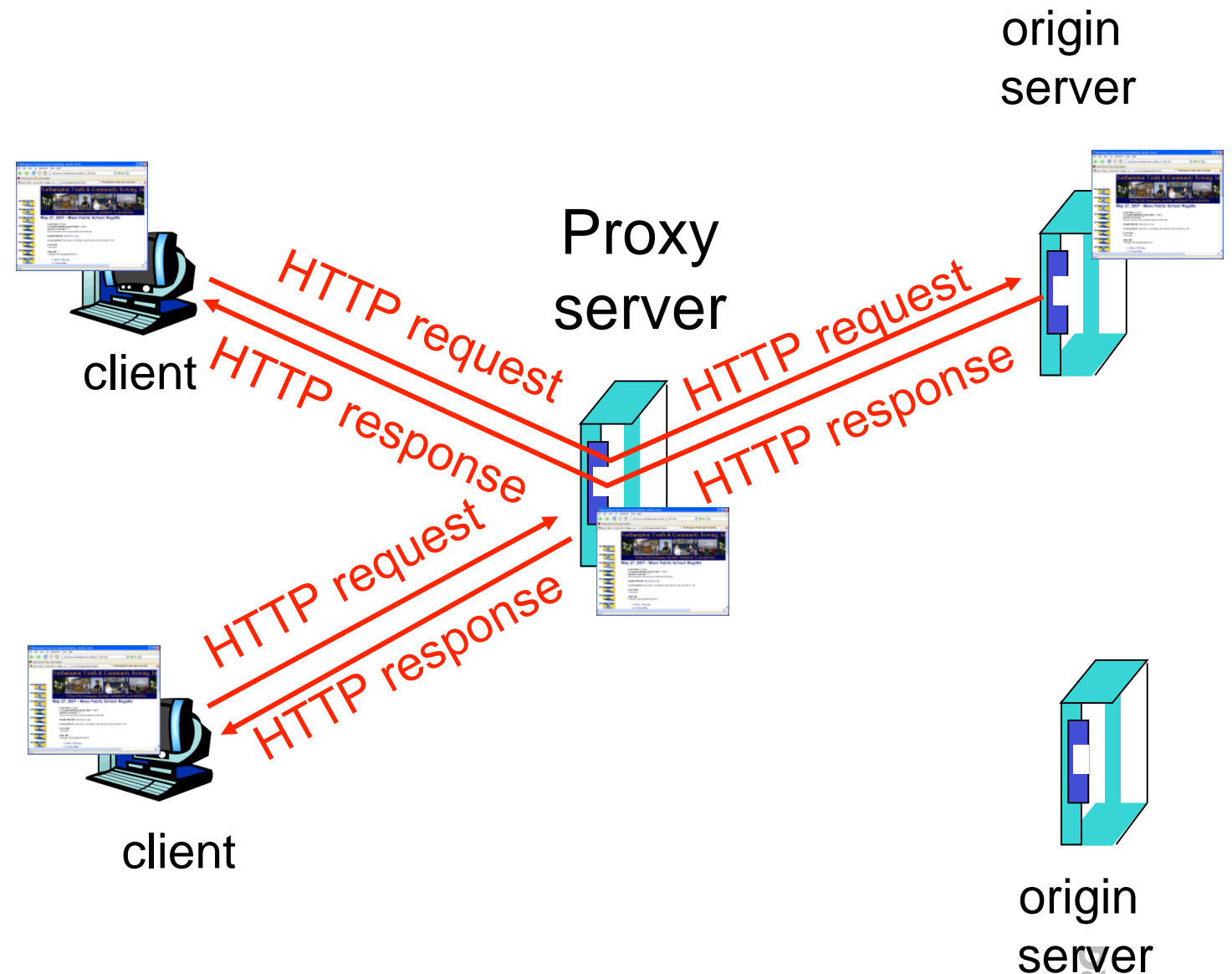


- Cookies erlauben
 - Authentifikation
 - Einkaufswagen
 - Empfehlungen
 - Sitzungs-Status des Benutzers (Web Mail)
- Wie man den Status unterhält
 - speichert Zustand zwischen verschiedenen Transaktionen
 - Cookies: HTTP Nachrichten transportieren den Status
- Cookies und Privatsphäre
 - Cookies übergeben der Web-Site eine Menge von Informationen
 - z.B. Name, E-Mail, Kaufverhalten, etc.

Web Caches (Proxy Server)

- Ziel:
 - Client-Anfragen erfüllen ohne den Original-Server zu verwenden
- Benutzer greift auf das Web per Cache zu
 - Hierfür wird Browser konfiguriert
- Browser sendet alle HTTP-Anfragen zum Cache
 - Ist das Objekt im Cache, dann wird das Objekt geliefert
 - ansonsten liefert der Original-Server an den Proxy-Server
 - dieser liefert dann das Objekt an den Client



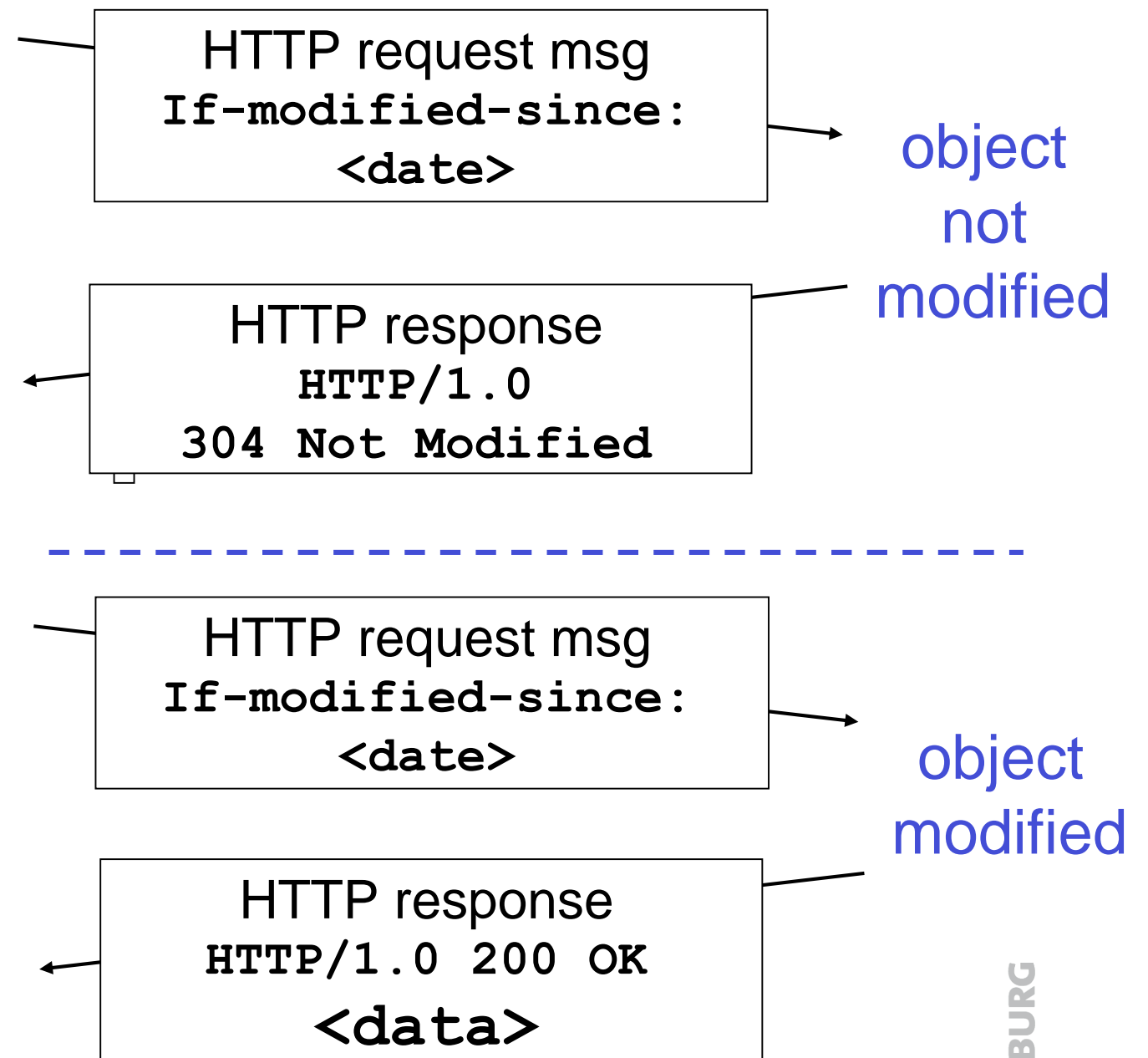
- Cache fungiert als Client und Server
 - typisch wird der Cache vom ISP (Internet Service Provider) bereit gestellt
- Warum
 - reduziert Antwortzeit für Client-Anfragen
 - reduziert den Verkehr über die Leitungen zu anderen ISPs
 - ermöglicht „kleinen“ Web-Servern effizient Inhalte zu verteilen

Conditional GET

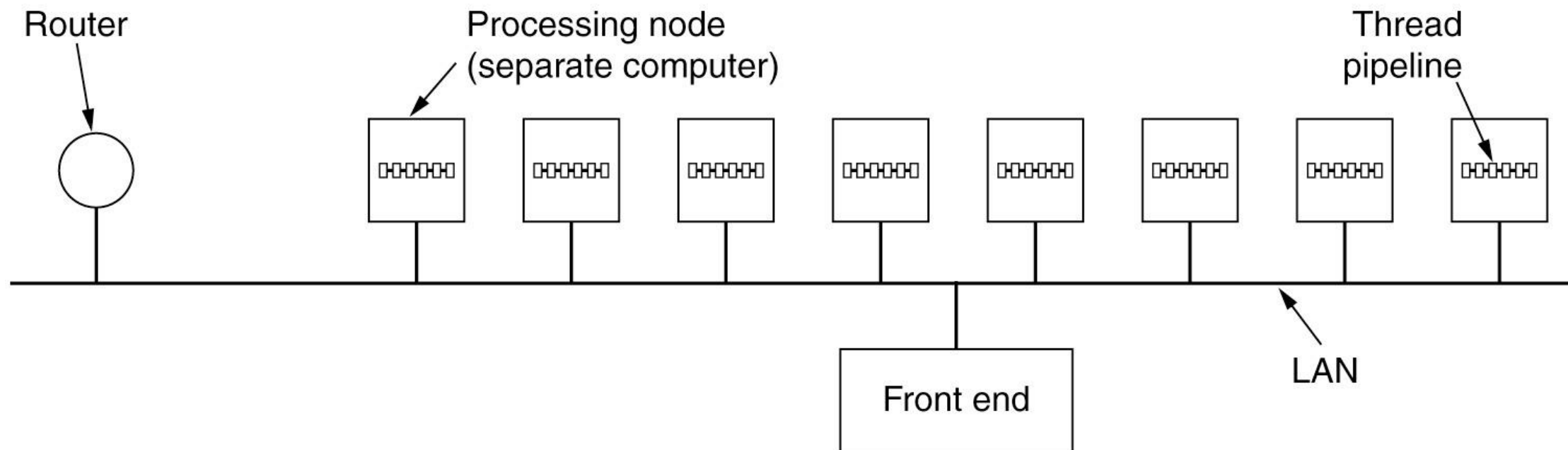
- Ziel: Objekt soll nicht gesendet werden, falls der Cache die aktuelle Version hat
- Cache: gibt den Zeitsempel der gecachten Kopie einer HTTP-Anfrage
 - If-modified-since: <date>
- Server: Antwort enthält kein Objekt, falls, die gecachte Kopie aktuell ist
 - HTTP/1.0 304 Not Modified

Cache

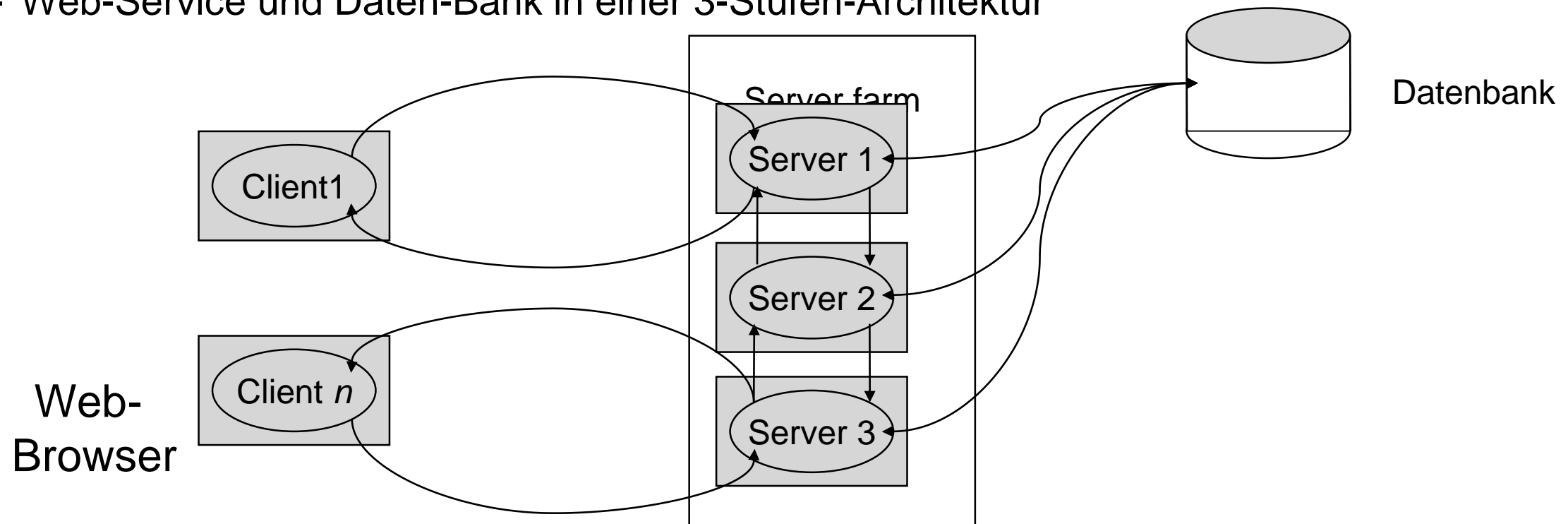
Server



- Um die Leistungsfähigkeit auf der Server-Seite zu erhöhen
 - wird eine Reihe von Web-Servern eingesetzt
- Front end
 - nimmt Anfragen an
 - reicht sie an separaten Host zur Weiterbearbeitung weiter



- Web-Server stellen nicht nur statische Web-Seiten zur Verfügung
 - Web-Seiten werden auch automatisch erzeugt
 - Hierzu wird auf eine Datenbank zurückgegriffen
 - Diese ist nicht statisch und kann durch Interaktionen verändert werden
- Problem:
 - Konsistenz
- Lösung
 - Web-Service und Daten-Bank in einer 3-Stufen-Architektur



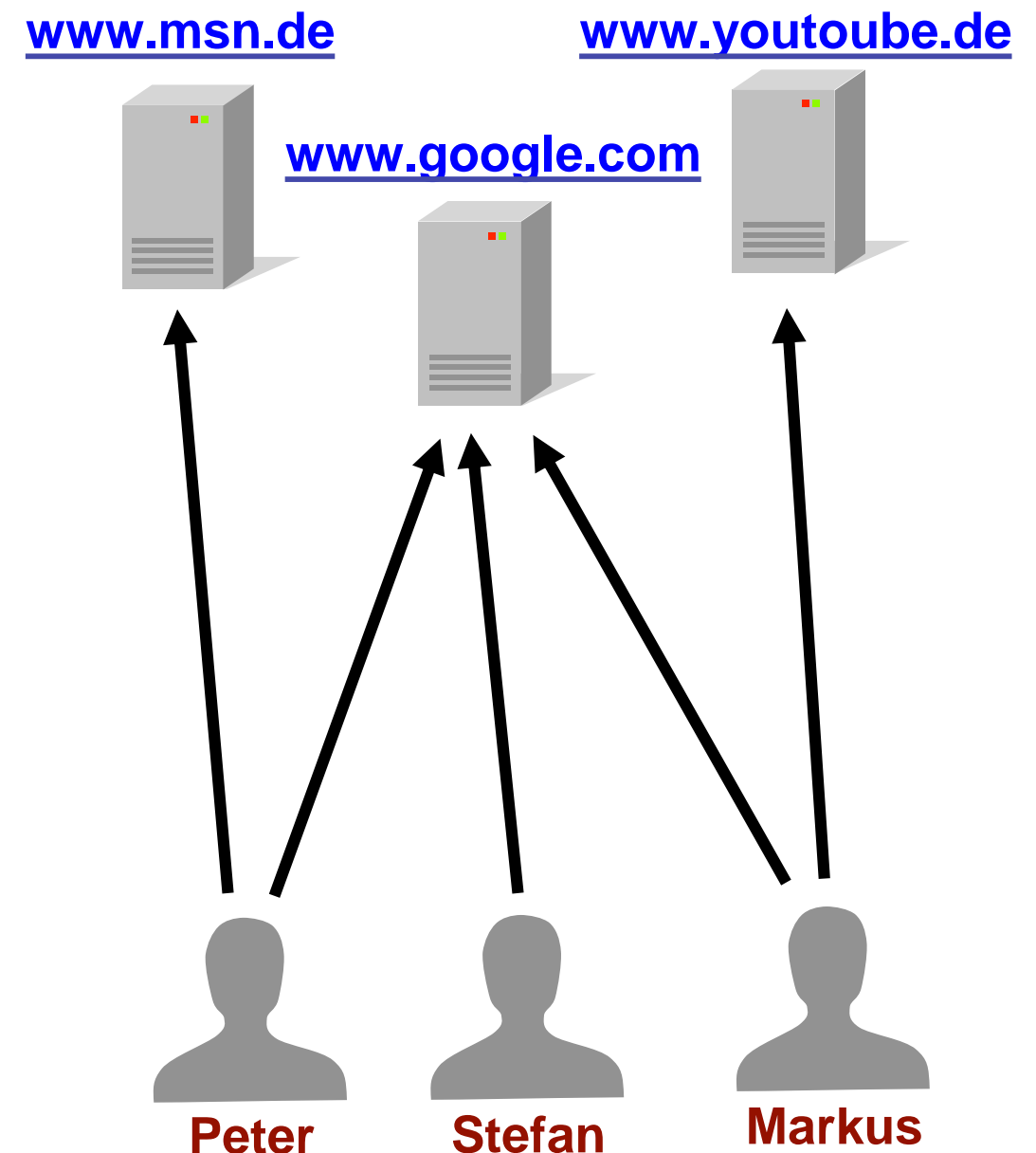
Beispiel: Google Data Centers

- Kosten eines Daten-Centers: 600 Mio US\$
- Google investierte 2007 2,4 Mrd. US\$ in Daten-Center
- Jedes Daten-Center verbraucht 50-100 MW

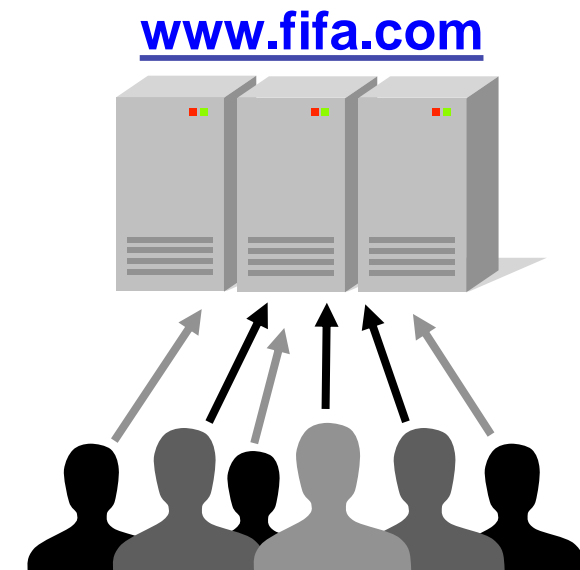


- Eine koordinierte Menge von Caches
 - Die Last großer Web-Sites wird verteilt auf global verteilte Server-Farmen
 - Diese übernehmen Web-Seiten möglichst verschiedener Organisationen
 - z.B. News, Software-Hersteller, Regierungen
 - Beispiele: Akamai, Digital Island
 - Cache-Anfragen werden auf die regional und lastmäßig bestgeeigneten Server umgeleitet
- Beispiel Akamai:
 - Durch verteilte Hash-Tabellen ist die Verteilung effizient und lokal möglich

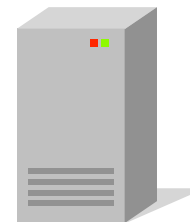
- Für Surfen im Web typisch:
 - Web-Server bieten Web-Seiten an
 - Web-Clients fordern Web-Seiten an
- In der Regel sind diese Mengen disjunkt
- Eingehende Anforderungen belasten Web-Server hinsichtlich:
 - Übertragungsbandbreite
 - Rechenaufwand (Zeit, Speicher)



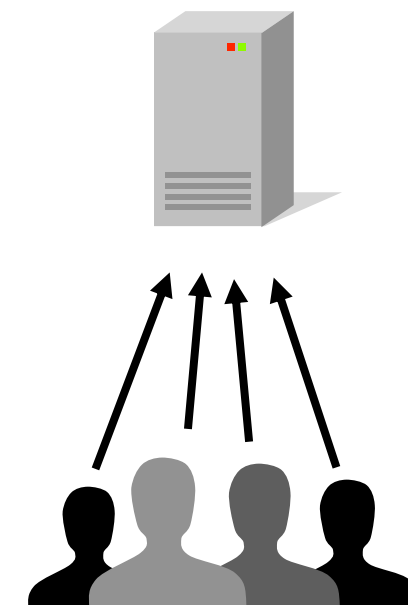
- Einige Web-Server haben immer hohe Lastanforderungen
 - Z.B. Nachrichten-Sites, Suchmaschinen, Web-verzeichnisse
 - Für permanente Anforderungen müssen Server entsprechen ausgelegt werden
- Andere leiden unter hohen Fluktuationen
 - z. B. bei besonderen Ereignissen:
 - fifa.com (Fussball-EM)
 - t-mobile.de (iPhone 6 Einführung)
 - Server-Erweiterung nicht sinnvoll
 - Bedienung der Anfragen aber erwünscht



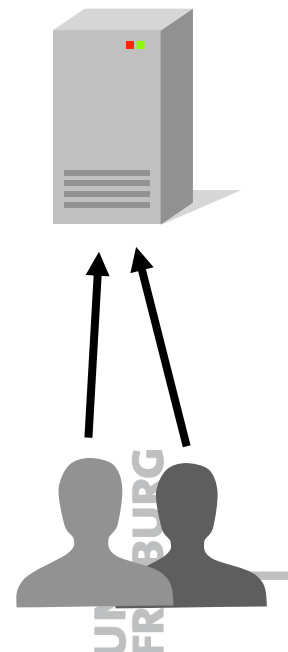
Montag



Dienstag

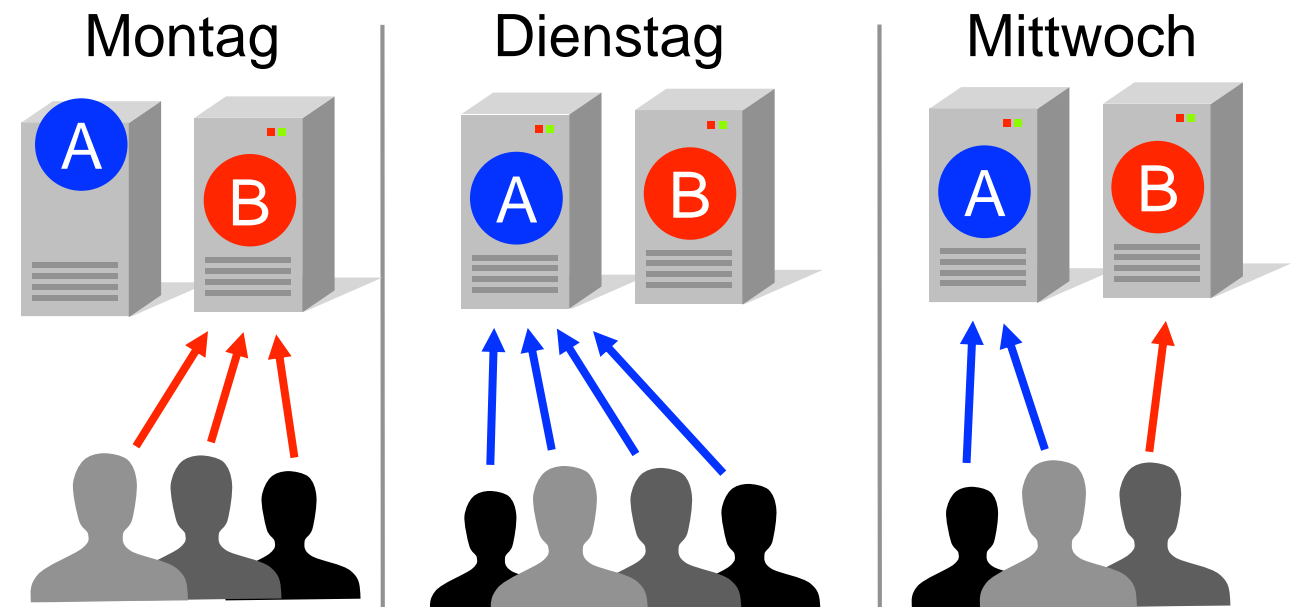


Mittwoch

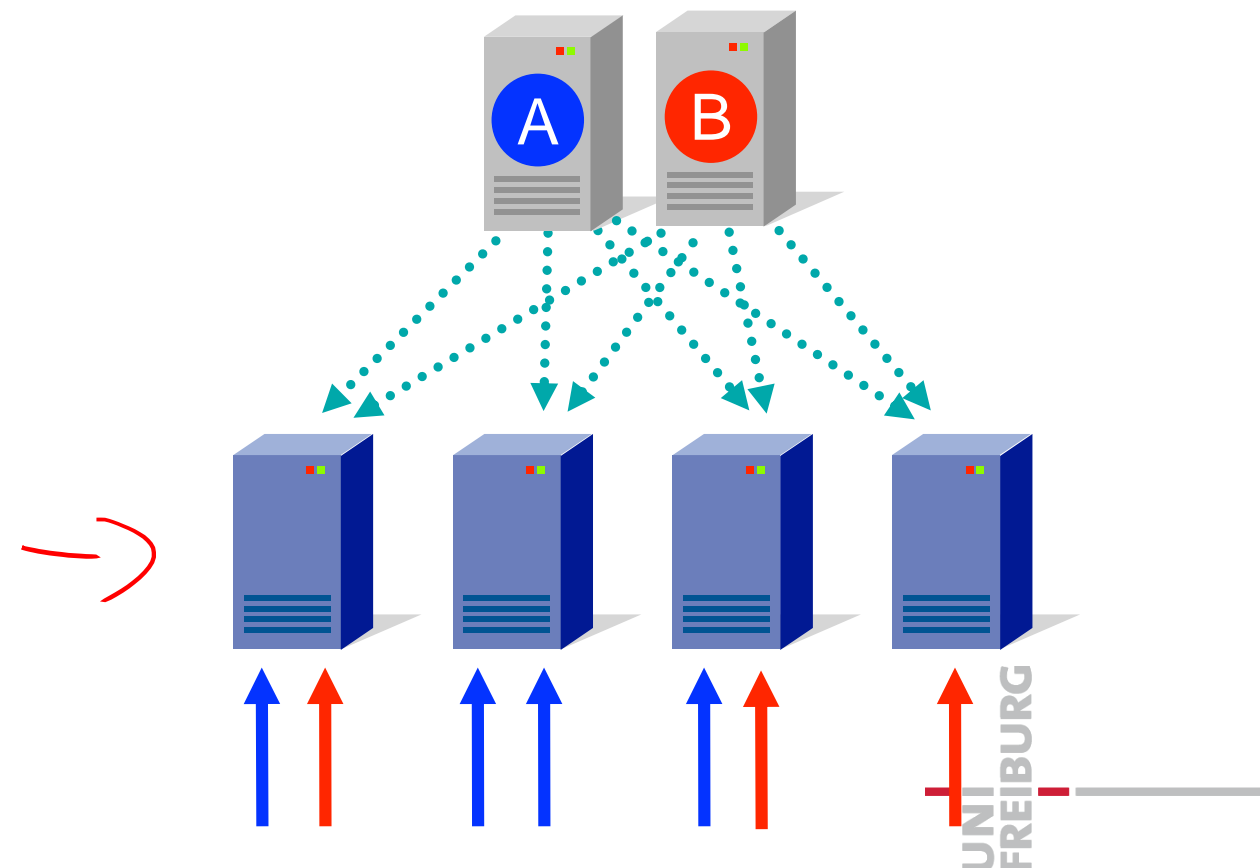


Lastbalancierung im WWW

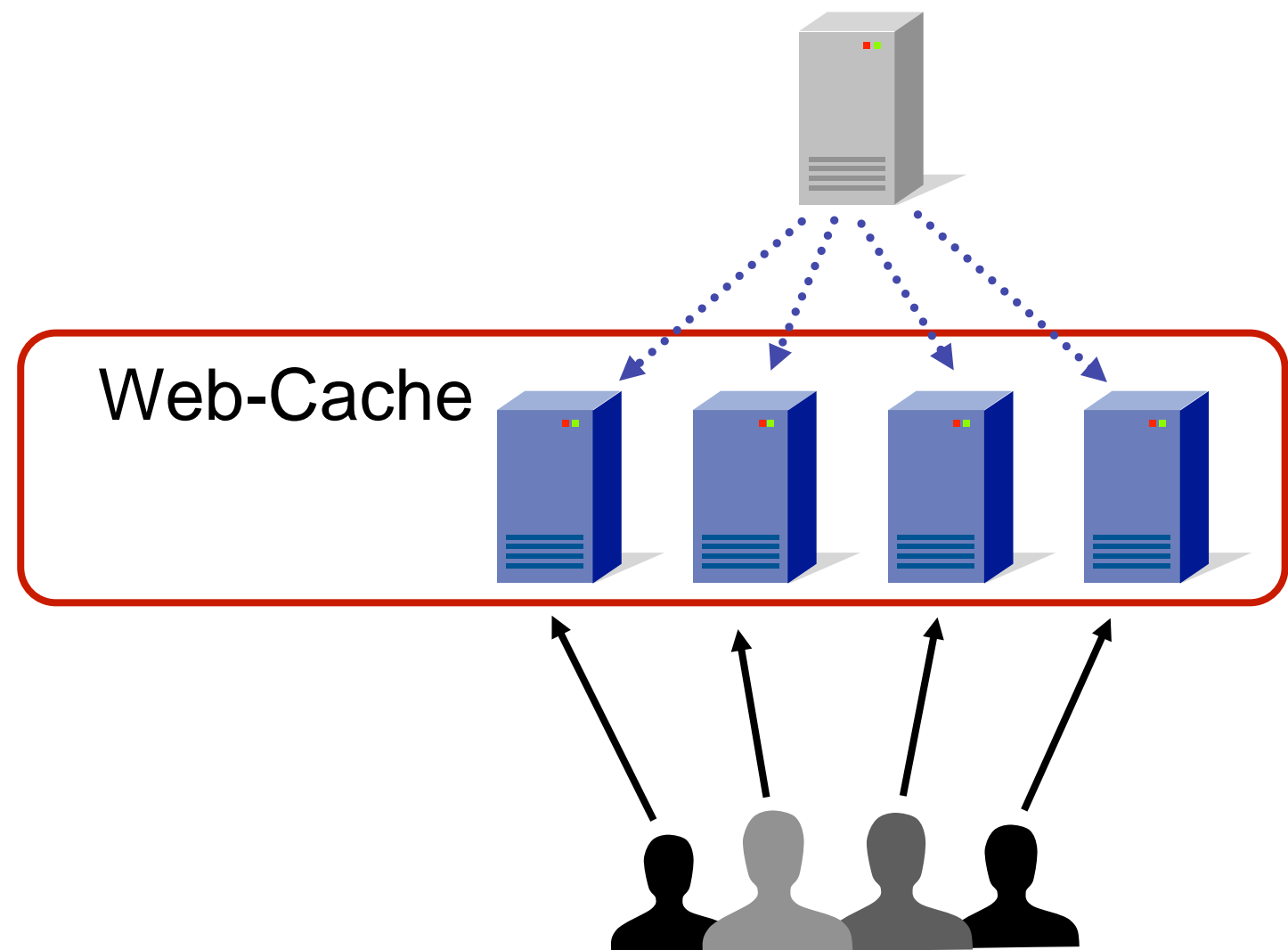
- Fluktuationen betreffen meistens einzelne Server



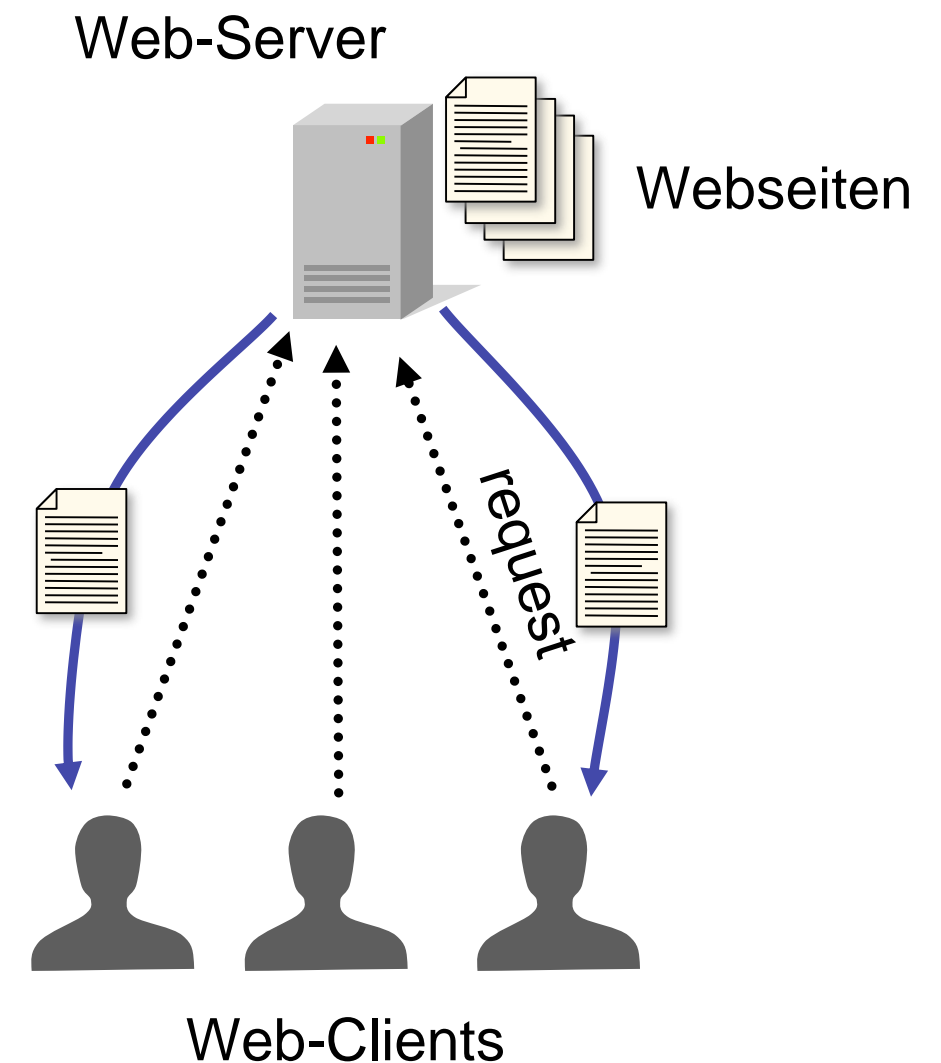
- (Kommerzielle) Lösung
 - Dienstleister bieten Ausweich-(Cache-)Server an
 - Viele Anforderungen werden auf diese Server verteilt
- Aber wie?



- Leighton, Lewin, et al.
STOC 97
 - *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web*
 - Passen bestehende Verfahren für dynamische Hash-Funktionen an WWW-Anforderungen an
- Leighton und Lewin (MIT) gründen Akamai 1997

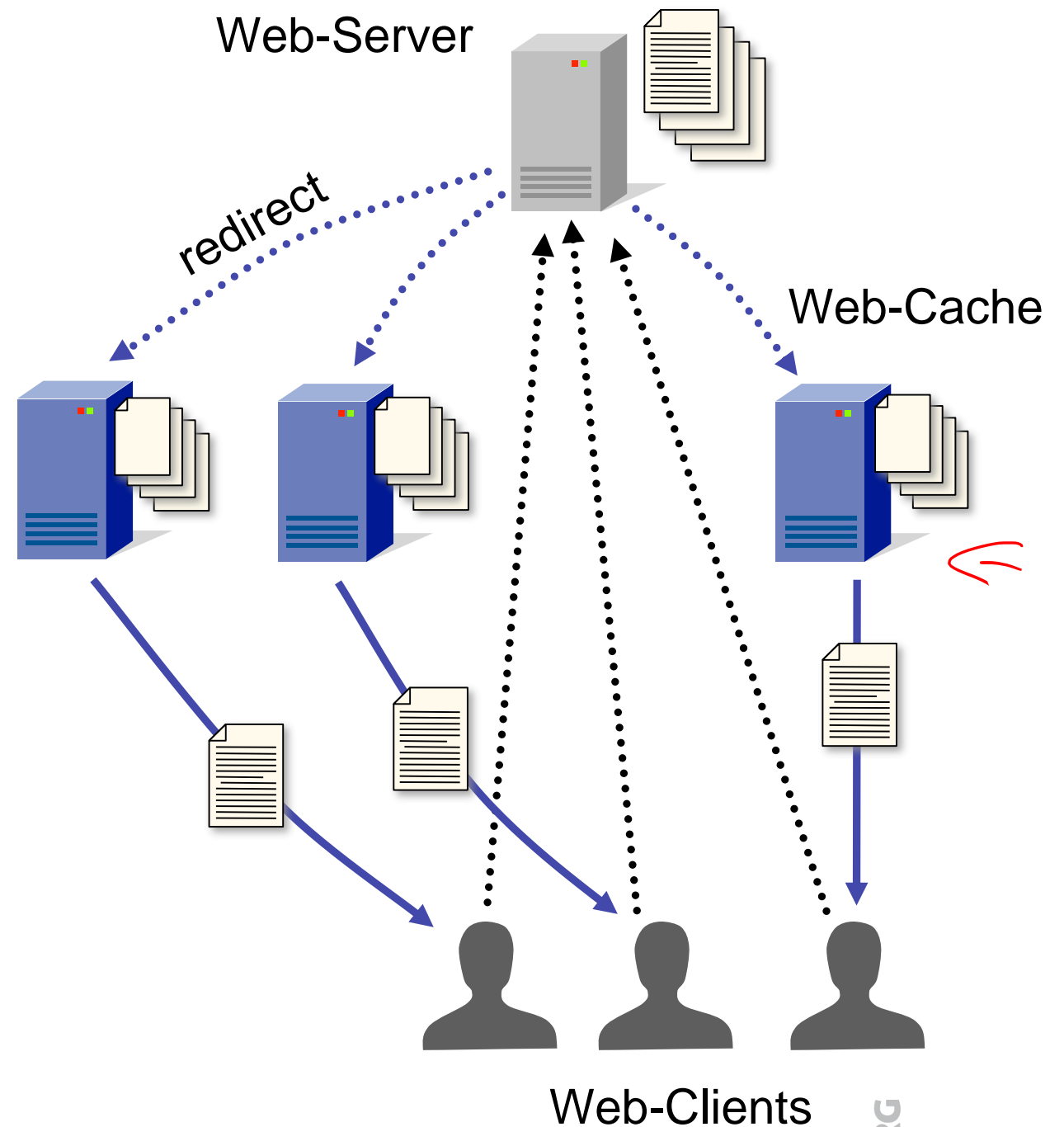


- Ohne Lastbalancierung:
 - Jeder Browser (Web-Client) belegt einen Web-Server für eine Web-Site
- Vorteil:
 - Einfach
- Nachteil:
 - Der Server muss immer für den Worst-Case ausgelegt werden



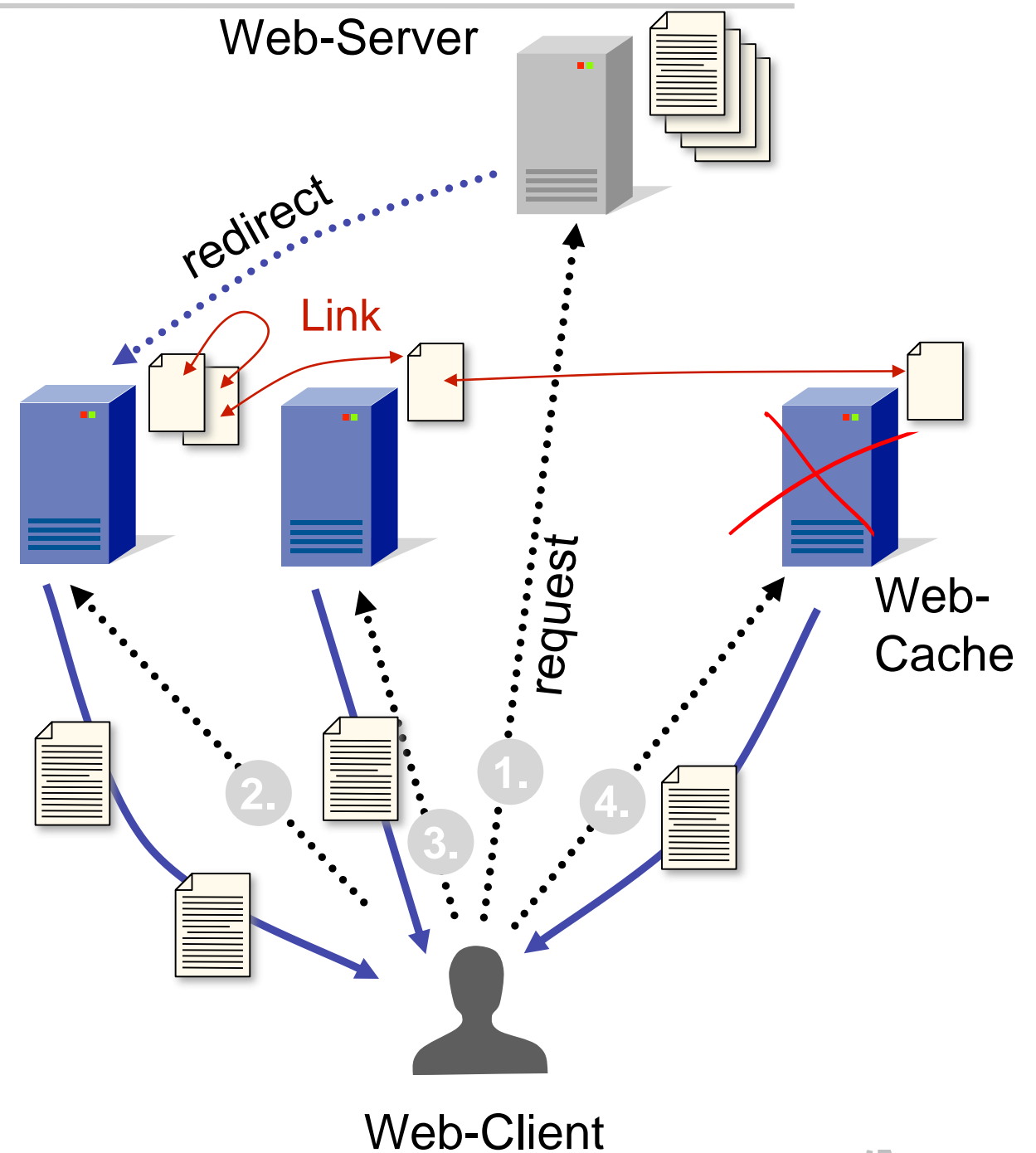
Site Caching

- Ganze Web-Site wird auf verschiedene Web-Caches kopiert
- Browser fragt bei Web-Server nach Seite
- Web-Server leitet Anfrage auf Web-Cache um (redirect)
- Web-Cache liefert Web-Seite aus
- Vorteil:
 - Gute Lastbalancierung für Seitenverteilung
- Nachteil:
 - Bottleneck: Redirect
 - Großer Overhead durch vollständige Web-Site-Replikationen



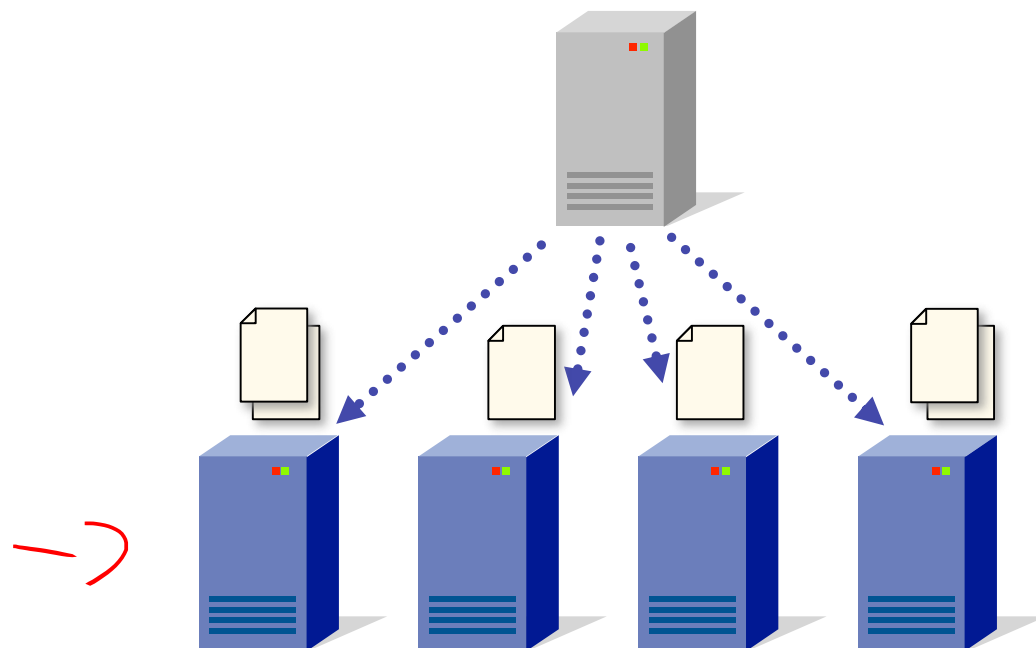
Proxy Caching

- Jede Web-Seite wird auf einige (wenige) Web-Caches verteilt
- Nur Startanfrage erreicht Web-Server
- Links referenzieren auf Seiten im Web-Cache
- Dann surft der Web-Client nur noch auf den Web-Cache
- Vorteil:
 - Kein Bottleneck
- Nachteil:
 - Lastbalancierung nur implizit möglich
 - Hohe Anforderung an Caching-Algorithmus

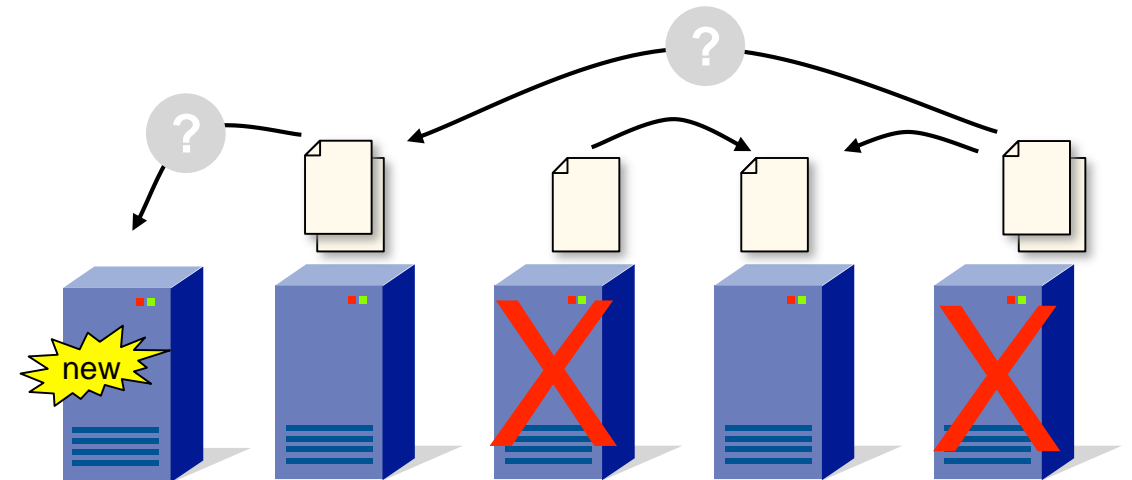


Anforderungen an Caching-Algorithmus

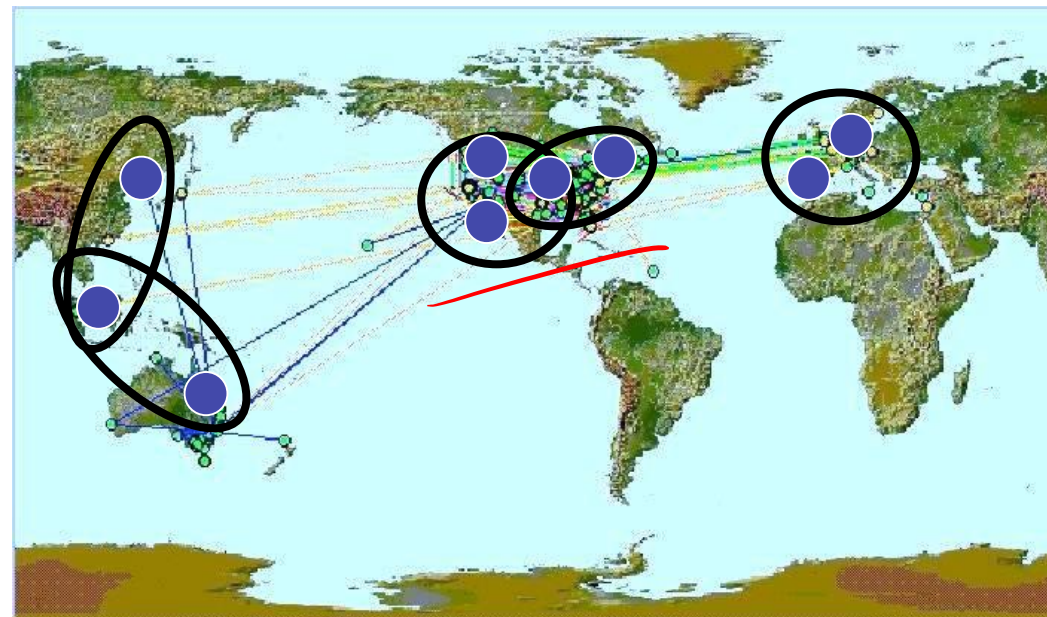
Balance
Gleichmäßige Verteilung der Seiten



Dynamik
Effizientes Einfügen/Löschen von neuen
Web-Cache-Servern

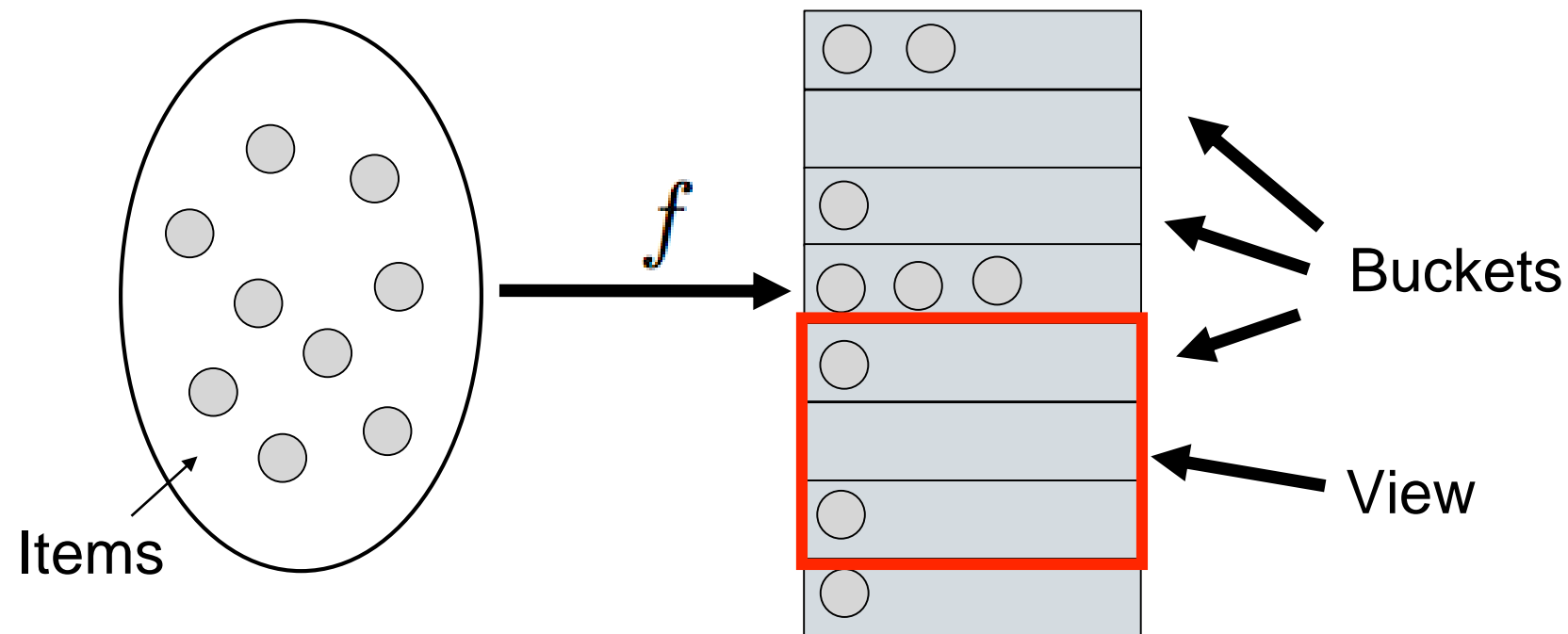
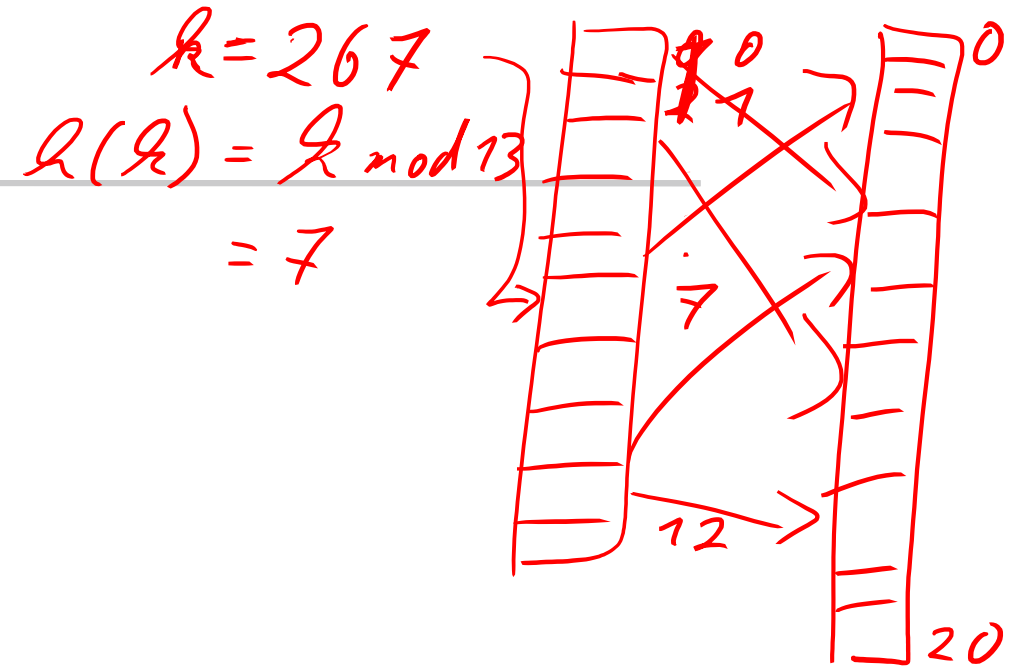


Views
Web-Clients „sehen“
unterschiedliche Menge
von Web-Caches

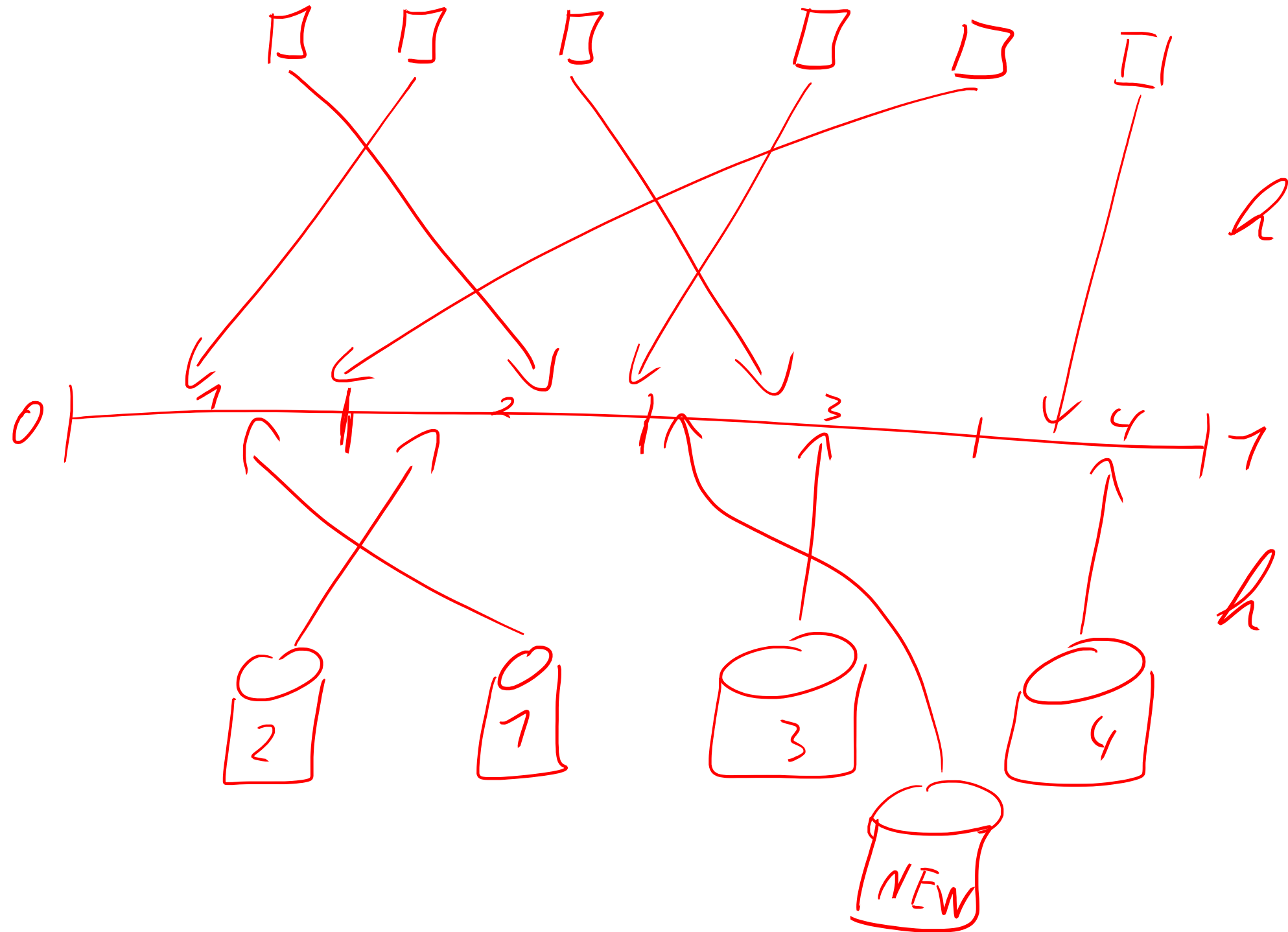


Ranged Hash-Funktionen oder Verteilte Hash-Funktionen

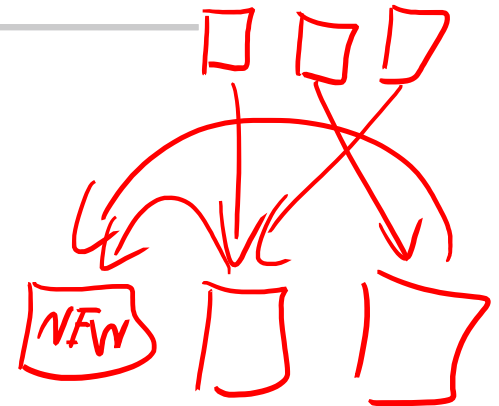
- Gegeben:
 - Elemente (Items)
 - Caches (Buckets)
 - Views: Menge von Caches
- Ranged Hash-Funktion:
 - Zuordnung eines Elements zu einem Cache in einem View



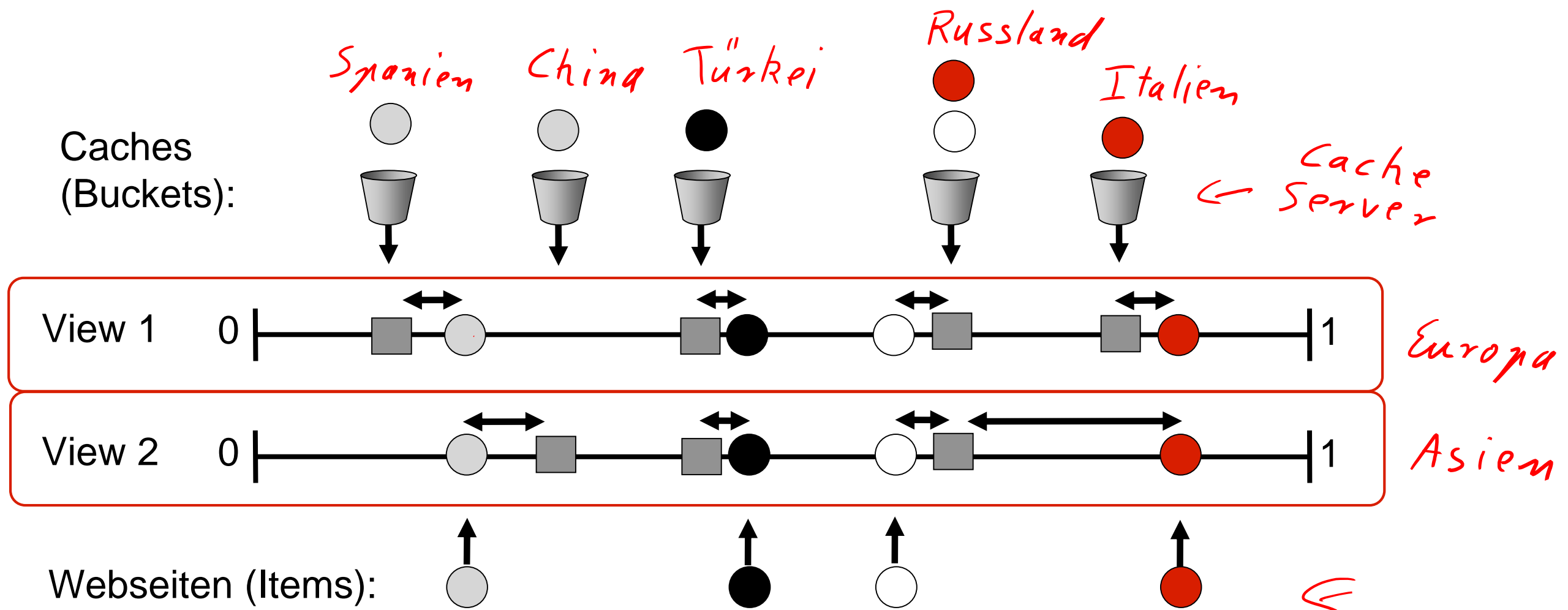
Distributed Hash Table



- Monotonie
 - nach dem Hinzufügen neuer Caches (Buckets) sollten keine Seiten (Items) zwischen alten Caches verschoben werden
- Balance
 - Alle Caches sollten gleichmäßig ausgelastet werden
- Spread (Verbreitung, Streuung)
 - Eine Seite sollte auf eine beschränkte Anzahl von Caches verteilt werden
- Load
 - Kein Cache sollte wesentlich mehr als die durchschnittliche Anzahl von Seiten enthalten



Distributed Hash Tables als Lösung



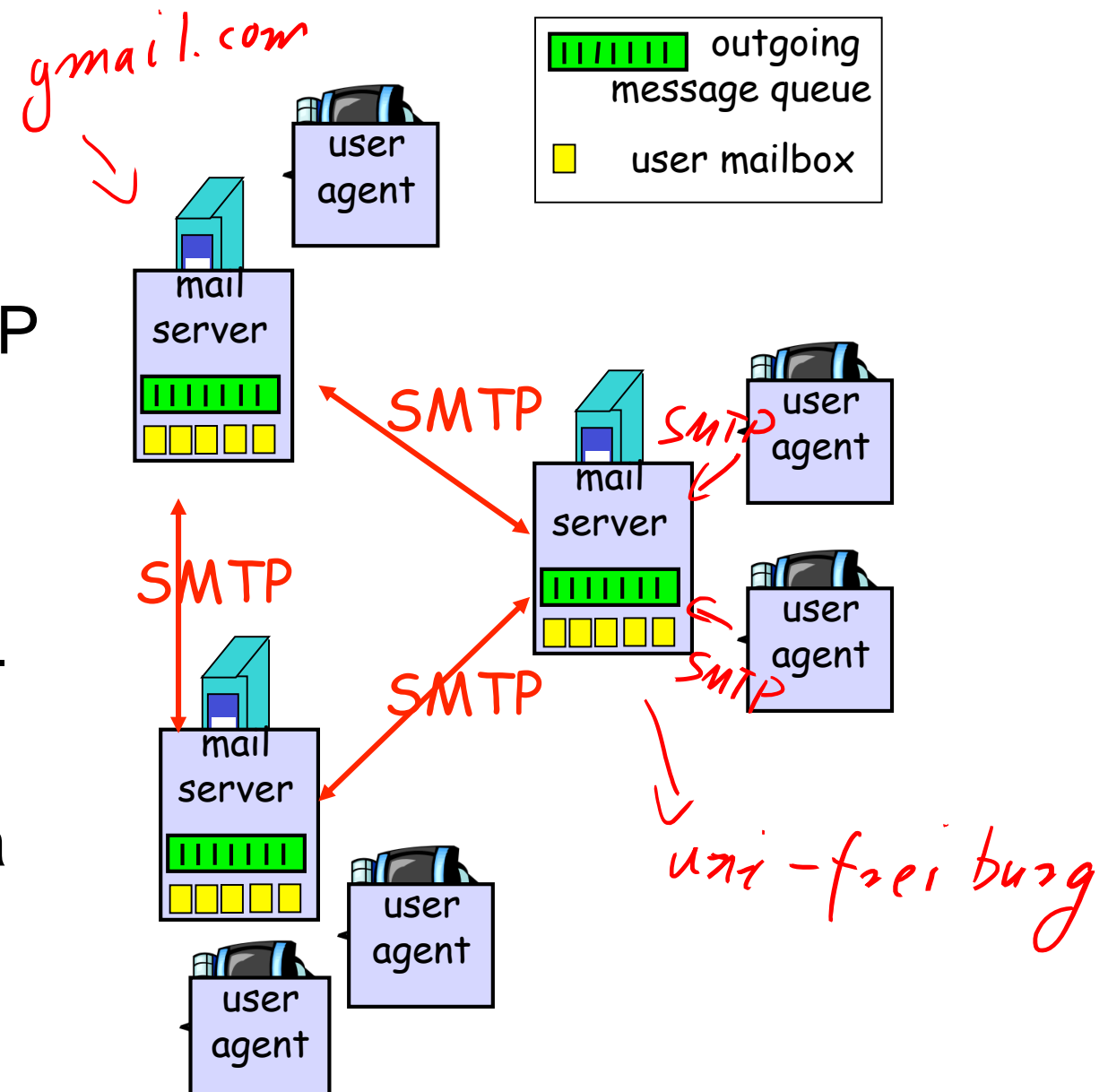
Electronic Mail

■ Hauptkomponenten

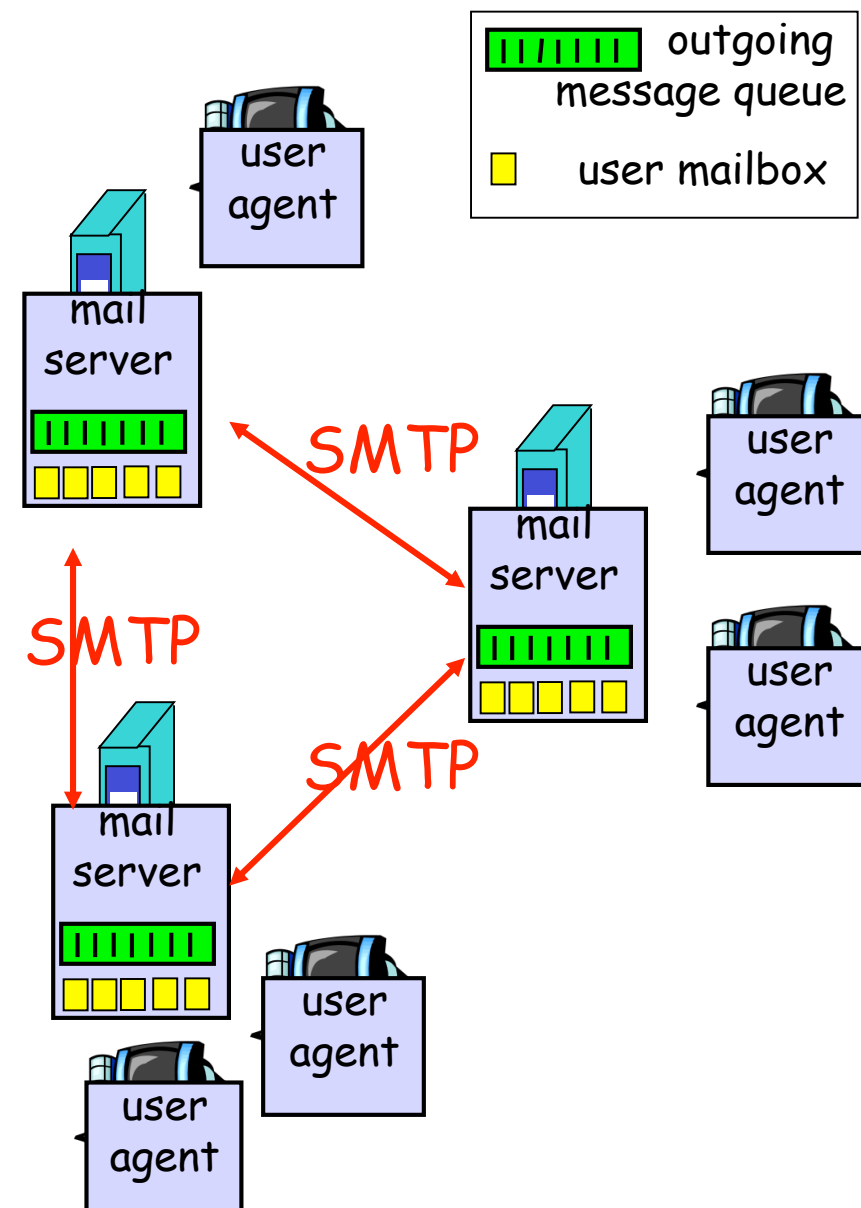
- user agents
- mail servers
- simple mail transfer protocol: SMTP

■ User Agent

- Mail Client
- Erstellen, ändern und lesen von E-Mail-Nachrichten
- z.B. Eudora, Outlook, pine, Mozilla Thunderbird
- abgehende und ankommende Nachrichten werden auf dem Server gespeichert



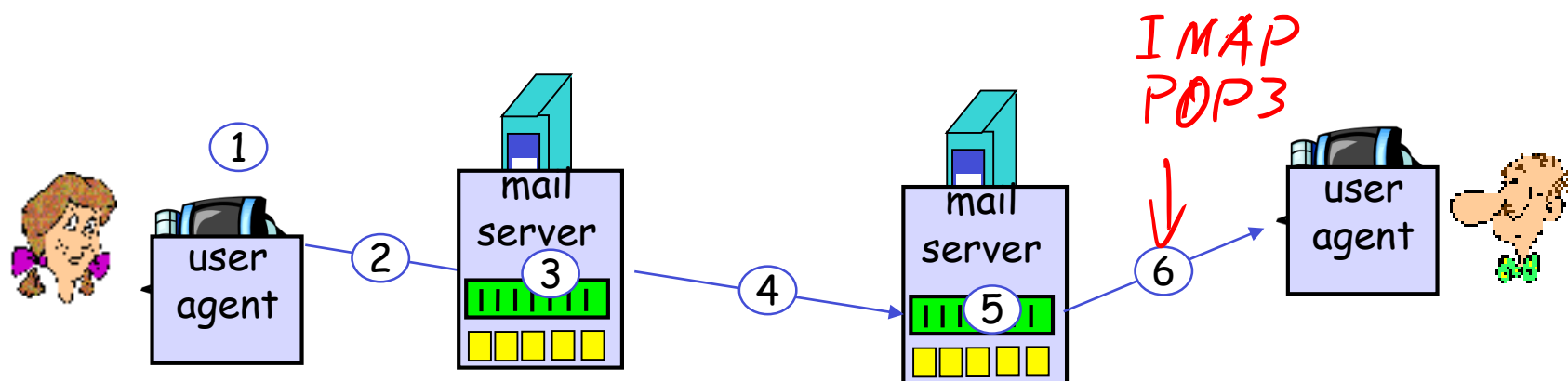
- Mailbox speichert eingehende Nachrichten für den User
- Nachrichten-Warteschlange (queue) der zu versendenden Nachrichten
- SMTP-Protocol zwischen Mail-Servern um E-Mail-Nachrichten zu schicken



- verwendet TCP um zuverlässig E-Mail-Nachrichten vom Client auf Port 25 zu verschicken
- Direkte Übertragung von Absender-Server zum Empfangs-Server
- 3 Phasen in der Übertragung
 - Handshake
 - Transfer der Nachricht
 - Abschluss
- Befehle und Antwort
 - Befehle als ASCII text
 - Antwort: Status-Code und Kurzbeschreibung
- Nachrichten sind in 7-bit ASCII

Beispiel: Alice sendet eine Nachricht an Bob

- 1) Alice verwendet UA um die Nachricht zu erzeugen mit Eintrag “to” bob@some school.edu
- 2) Alice UA sendet die Nachricht zu ihren Mail-Server
 - Nachricht wird in der Nachrichtenwartenschlange platziert
- 3) Client-Seite des SMTP öffnet TCP-Verbindung mit Bobs Mail-Server
- 4) SMTP Client sendet Alice Nachricht über die TCP-Verbindung
- 5) Bobs Mail-Server schreibt die Nachricht in Bobs Mailbox
- 6) Bob ruft seinen User Agent auf, um die Nachricht zu lesen

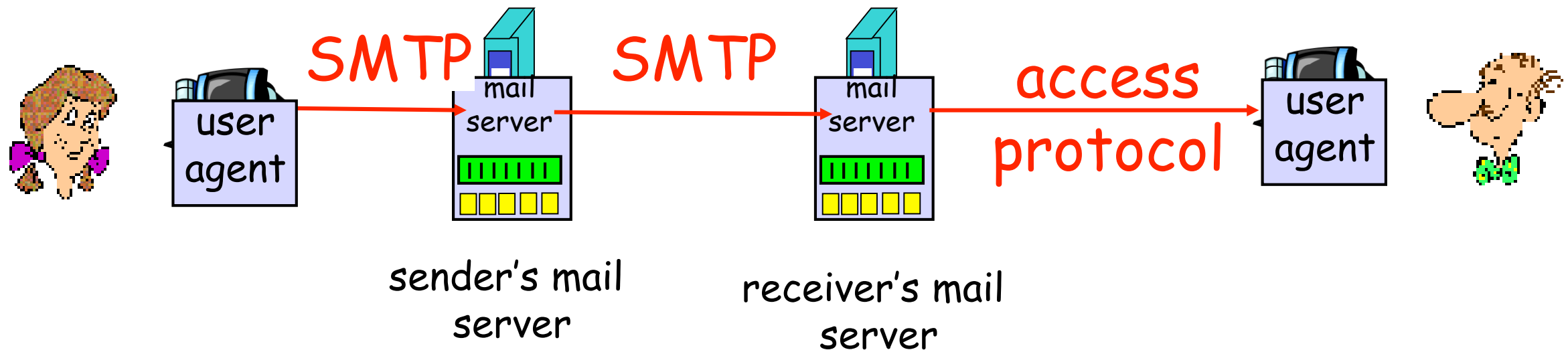


Beispiel SMTP Interaktion

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

- SMTP
 - verwendet persistente Verbindungen
 - verlangt Nachrichten (header & body) in 7-bit ASCII
 - SMTP-Server verwenden „CRLF.CRLF“ um das Ende einer Nachricht zu beschreiben
- Vergleich mit HTTP:
 - HTTP: pull
 - SMTP: push
 - beide haben ASCII Befehls- und Antwort-Interaktion und Status-Codes
- HTTP
 - jedes Objekt wird in eigener Nachricht verpackt
- SMTP
 - verschiedene Objekte werden in einer Multipart-Nachricht verschickt

Mail-Zugriffsprotokolle

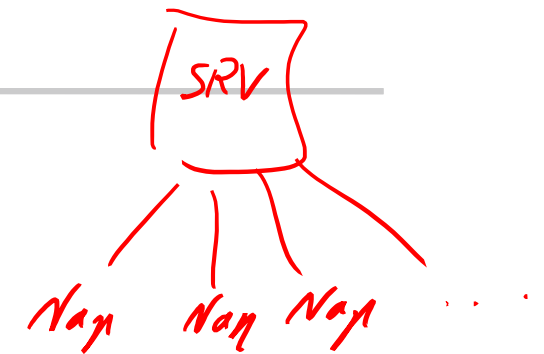


- SMTP: Auslieferung und Speicher zum Server des Empfängers
- Mail-Zugriffsprotokolle: E-Mail-Abruf vom Server
 - POP: Post Office Protocol [RFC 1939]
 - Authentifizierung (zwischen Agent und Server) und Download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - mehr Features und komplexer
 - Bearbeitung von gespeicherten Nachrichten **auf** dem Server
 - HTTP: gmail, Hotmail, Yahoo! Mail, web.de, etc.

round cube

- POP3 (Post-Office-Protocol)
 - User kann im “download and delete” Modus E-Mails einmalig herunterladen
 - User kann E-Mails noch einmal lesen, wenn er den Client wechselt:
 - “Download-and-keep”: Kopien der Nachricht auf verschiedenen Clients
 - POP3 ist zustandslos (stateless) von einer Sitzung zur nächsten
- IMAP (Internet Message Access Protocol)
 - hält alle Nachrichten an einem Ort: dem Server
 - erlaubt dem User die Nachrichten in Ordnern zu organisieren
 - IMAP speichert den Benutzer-Status zwischen Sitzungen
 - Namen der Ordner und Zuordnung zwischen Nachrichten-ID und Ordnernamen

- Napster 1999-2000
 - Filesharing, nur rudimentäres P2P
- Gnutella 2000
 - 1. echtes P2P-Netzwerk
- Edonkey 2000
 - Mehr Filesharing als P2P
- FreeNet 2000
 - Anonymisiertes P2P-Netzwerk
- FastTrack 2001
 - KaZaa, Morpheus, Grokster
- Bittorrent 2001
- Skype 2003
 - VoIP (voice over IP), Chat, Video



- Distributed Hash-Tables (DHT) (1997)
 - Ziel: Lastbalancierung für Web-Server
- CAN (2001)
 - DHT-Netzwerk-Struktur
- Chord (2001)
 - Erstes effiziente P2P-Netzwerk
 - Logarithmische Suchzeit
- Pastry/Tapestry (2001)
 - Effizientes verteiltes P2P-Netzwerk unter Verwendung des Plaxton-Routing
- Und viele andere Ansätze
 - Viceroy, Distance-Halving, Koorde, Skip-Net, P-Grid, ...
- In den letzten fünf Jahren:
 - Network Coding for P2P
 - Game theory in P2P
 - Anonymity, Security

- Was ist P2P **NICHT**?
 - Ein Client-Server network
- Etymologie: peer
 - lateinisch: par = gleich
 - Standesgleich
 - P2P, Peer-to-Peer: Beziehung zwischen gleichwertigen Partnern
- Definition
 - Ein Peer-to-Peer Network ist ein Kommunikationsnetzwerk im Internet
 - ohne zentrale Kontrolle
 - mit gleichwertigen, unzuverlässigen Partnern

Distributed Hash-Table (DHT)

■ Hash-Tabellen

- nicht praktikabel in P2P

■ Verteilte Hash-Tabellen

- *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web*, Karger, Lehman, Leighton, Levine, Lewin, Panigrahy, STOC 1997

■ Daten

- werden *gehasht* und nach Bereich den Peers zugeordnet

■ Peers

- werden an eine Stelle *gehasht* und erhalten Bereiche des Wertebereichs der Hashfunktion zugeteilt

