Warstwa aplikacji

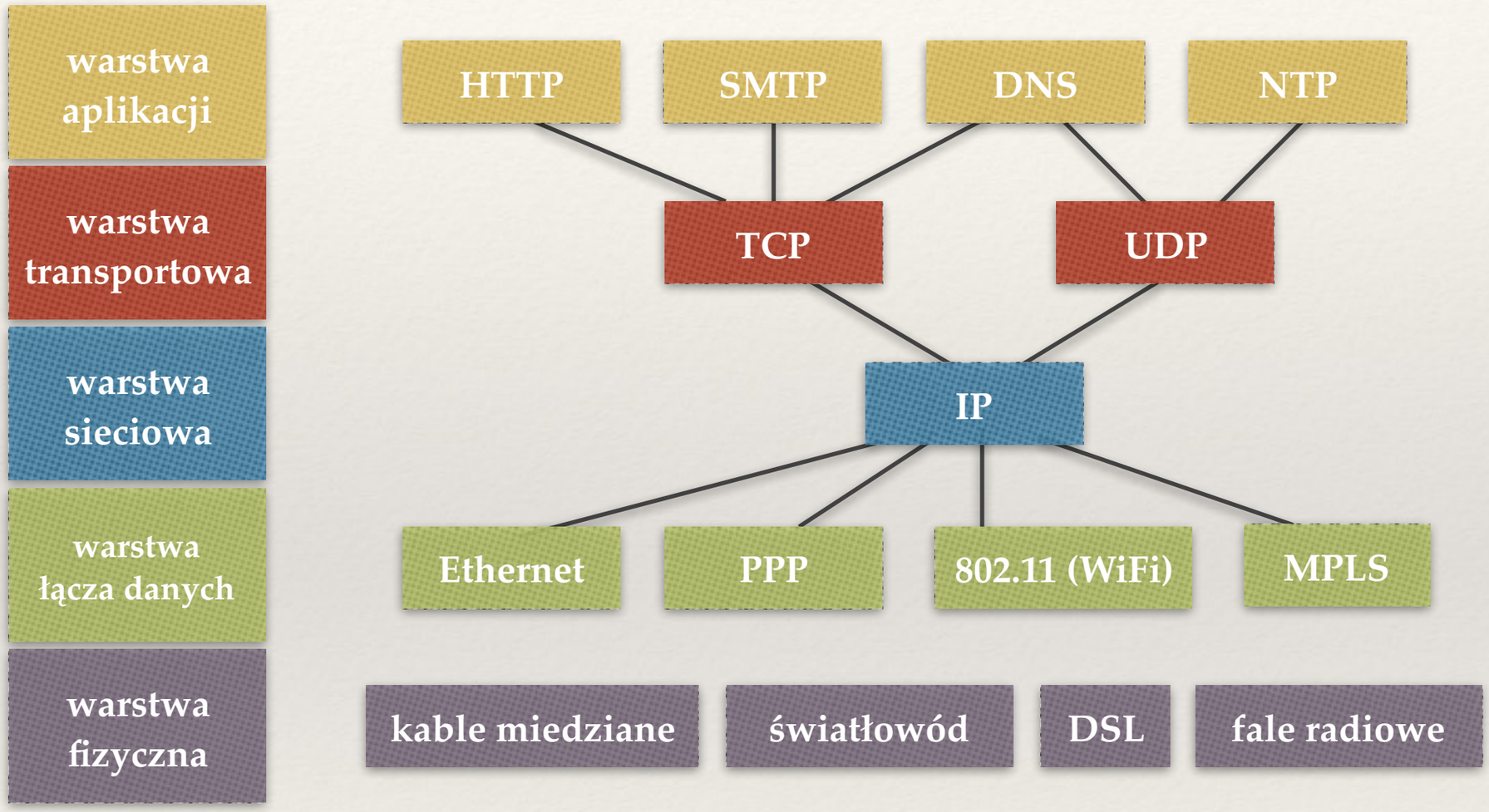
część 2

Sieci komputerowe

Wykład 8

Marcin Bieńkowski

Protokoły w Internecie



W dzisiejszym odcinku

- Poczta elektroniczna
- NAT vs. warstwa aplikacji
- Wydajność HTTP

Poczta elektroniczna

Protokół SMTP

- Protokół przekazywania poczty.
- Protokół czysto tekstowy, serwer nasłuchuje na porcie 25.

Wysyłanie bezpośrednie (1)

Chcemy wysłać pocztę do adresu `abc@xyx.com`.

- Łączymy się z adresem IP serwera odpowiedzialnego za odbieranie i przechowywanie pocztę dla domeny `xyx.com`.

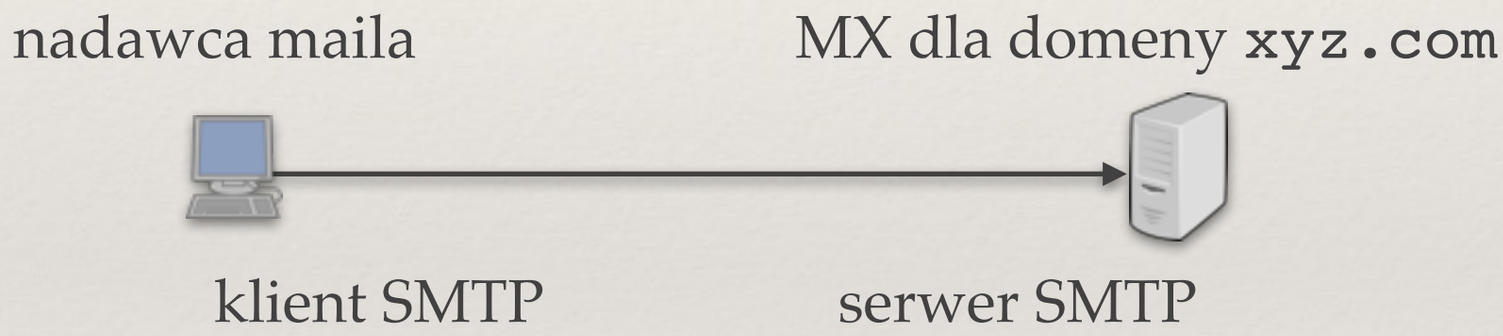
- Rekord MX (*mail exchange*) w DNS a rekord A:

```
ii.uni.wroc.pl                 A    156.17.4.11
ii.uni.wroc.pl                 MX    swiatowit.ii.uni.wroc.pl.
swiatowit.ii.uni.wroc.pl.      A    156.17.4.1
```

Wysyłając pocztę do `abc@ii.uni.wroc.pl` łączymy się z `156.17.4.1` (nie z `156.17.4.11`).

Wysyłanie bezpośrednie (2)

Chcemy wysłać pocztę do adresu `abc@xyx.com`.



Wysyłanie pośrednie

Chcemy wysłać pocztę do adresu `abc@xyx.com`.

Krok 1.

nadawca maila — SMTP relay / smarthost

klient SMTP — serwer SMTP

= to co program pocztowy klienta ma ustawione jako „serwer SMTP“

Krok 2.

SMTP relay / smarthost — MX dla domeny `xyz.com`

klient SMTP — serwer SMTP

(Kroków może być więcej).

Przekazywanie pośrednie

- Zazwyczaj wymaga autoryzacji nadawcy u SMTP relay
  - Różne mechanizmy autoryzacji są elementem protokołu SMTP.
  - Zabezpieczenie przed rozsyłaniem niechcianej poczty (spamu).
  - Czasem autoryzacja na podstawie adresu IP klienta.

Przykładowy email (otrzymany)

```
Delivered-To: marcin.bienkowski@cs.uni.wroc.pl
Received: by 10.64.232.142 with SMTP id to14csp146725iec;
        Sat, 23 Apr 2016 08:41:37 -0700 (PDT)
Received: from aisd.ii.uni.wroc.pl (algo2014.ii.uni.wroc.pl. [156.17.4.30])
        by mx.google.com with ESMTP id l199si7484936lfl.24.2016.04.23.08.41.36
        for <marcin.bienkowski@cs.uni.wroc.pl>;
        Sat, 23 Apr 2016 08:41:36 -0700 (PDT)
Received: by aisd.ii.uni.wroc.pl (Postfix, from userid 1000) id E6BCD5F84D;
        Sat, 23 Apr 2016 17:41:35 +0200 (CEST)
Date: Sat, 23 Apr 2016 17:41:35 +0200
From: mbi <mbi@ii.uni.wroc.pl>
To: marcin.bienkowski@cs.uni.wroc.pl
Subject: Testowy email
Message-ID: <20160423154135.GA11834@aisd.ii.uni.wroc.pl>
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Disposition: inline
Content-Transfer-Encoding: 8bit
User-Agent: Mutt/1.5.23 (2014-03-12)


Jakaś treść maila.
M.
```

pola ustawiane przez odbiorcę

pola ustawiane przez serwery pośredniczące

pola ustawiane przez nadawcę

Pola nagłówka ustawiane przez klienta

- From:
- To:
- Subject:
- Cc:
- Bcc: („ślepa kopia“)
- Message-ID: (unikatowy identyfikator wiadomości)
- Date: (data wysłania)
- In-Reply-To: (ID maila, na którego odpowiadamy)
- References:

Typ zawartości

Pole `Content-Type:` nagłówka określa:

- czym jest treść maila (w standardzie MIME)
  - czysty tekst (`text/plain`)
  - HTML (`text/html`)
- kodowanie znaków

```
Content-Type: text/plain; charset=utf-8

Content-Transfer-Encoding: 8bit
```

Załączniki pocztowe

```
Content-Type: multipart/mixed; boundary=„--UNIKATOWY-CIĄG-ZNAKÓW“
Content-Transfer-Encoding: 8bit

--UNIKATOWY-CIĄG-ZNAKÓW
Content-Type: text/plain; charset=utf-8
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Wiadomość testowa
M.

--UNIKATOWY-CIĄG-ZNAKÓW
Content-Type: image/jpeg
Content-Disposition: attachment; filename="obrazek.jpg"
Content-Transfer-Encoding: base64

ZAWARTOŚĆ-PLIKU-ZAKODOWANA-W-BASE64.

--UNIKATOWY-CIĄG-ZNAKÓW
```

treść tekstowego maila w UTF-8

załącznik obrazek.jpg

Dostarczanie poczty do użytkownika

- Protokół POP3.
- Protokół IMAP.
- Klienty pocztowe jako aplikacje WWW.

Spam: niechciane wiadomości pocztowe

Sposoby wykrywania i usuwania spamu:

- Ręczne blokowanie konkretnych tematów / nadawców
- Metody statystyczne (filtry bayesowskie)
- Greylisting
- SPF
- …

Filtry bayesowskie (1)

Założenia:

- Dostajemy losowy email (z przestrzeni wszystkich maili).
- Jest w nim słowo `viagra`.
- Jaka jest szansa, że ten email to spam?

$$\Pr[\texttt{spam} \mid \texttt{viagra}] = \frac{\Pr[\texttt{viagra} \cap \texttt{spam}]}{\Pr[\texttt{viagra}]}$$

$$= \frac{\Pr[\texttt{viagra} \mid \texttt{spam}] \cdot \Pr[\texttt{spam}]}{\Pr[\texttt{viagra}]}$$

Filtry bayesowskie (2)

- Bierzemy zbiór słów $\{W_i\}_i$
- Estymujemy dla każdego z nich $\Pr[W_i \mid \text{spam}]$ i $\Pr[W_i \mid \text{not spam}]$ na podstawie zbioru treningowego.
- Zakładamy niezależność $\Pr[W_i]$ i $\Pr[W_i \mid \text{spam}]$.

$$\Pr[\text{spam} \mid \cap_{i=1}^{k} W_i] = \frac{\Pr[\cap_{i=1}^{k} W_i \mid \text{spam}] \cdot \Pr[\text{spam}]}{\Pr[\cap_{i=1}^{k} W_i]}$$

$$= \frac{\Pi_{i=1}^{k} \Pr[W_i \mid \text{spam}] \cdot \Pr[\text{spam}]}{\Pi_{i=1}^{k} \Pr[W_i]}$$

Greylisting

Wolne wysyłanie spamu jest nieopłacalne.

- Pomysł: każemy klientowi SMTP wysłać wiadomość ponownie za jakiś czas:

```
 -> MAIL FROM: <nadawca@jakasdomena.pl>
<- 250 2.1.0 Sender ok
-> RCPT TO: <odbiorca@innadomena.pl>
<- 451 4.7.1 Please try again later
```

- Ustawiamy okno czasowe („nie wcześniej niż za 10 min. i nie później niż po godzinie“).
  - Jeśli klient ponowi w danym oknie czasowym, to akceptujemy jego email.
  - Problemy z poprawnym ustawieniem okna.
  - Dostarczanie poczty przestaje być szybkim procesem.

- Stosowany wariant: zamiast odrzucać, odbieraj wolniej z okna TCP.

Spam: SPF (Sender Policy Framework)

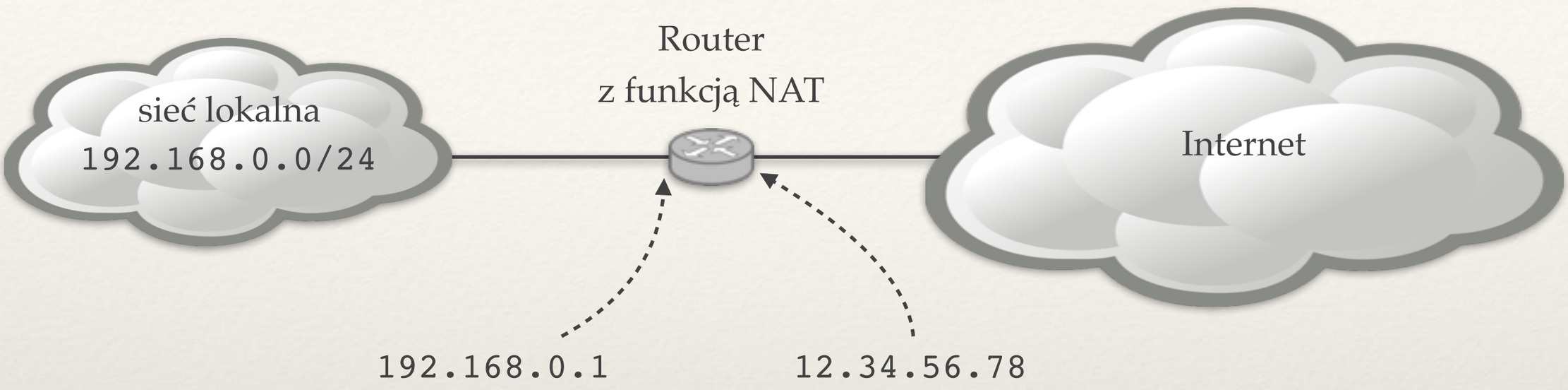
- Rekord SPF w DNS dla danej domeny:
  - `ii.uni.wroc.pl TXT "v=spf1`
    `ip4:156.17.4.0/24`
    `mx:ii.uni.wroc.pl`
    `mx:gmail.com`
    `mx:google.com`
    `-all"`

- Definiuje jakie komputery są uprawnione do wysyłania poczty z polem `From:` równym *`adres`*`@ii.uni.wroc.pl`.
  - komputery z adresów `156.17.4.0/24`.
  - serwery SMTP obsługujące pocztę dla domen `ii.uni.wroc.pl`, `gmail.com` i `google.com`.

- Rekord sprawdzany przez odbiorcę.
  - Problemy przy przekazywaniu poczty (komputer przekazujący nie jest już oryginalnym nadawcą!)

NAT vs. warstwa aplikacji

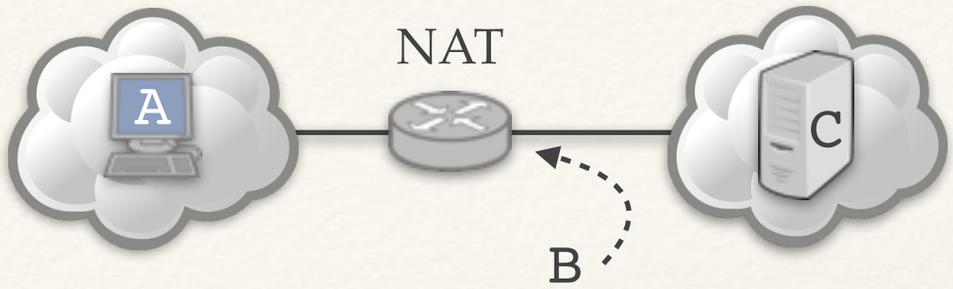
NAT



- Bardzo powszechne rozwiązanie.
- Z reszty Internetu cała sieć lokalna wygląda tak samo, jak pojedynczy komputer z adresem `12.34.56.78`.

Co robi router z funkcją NAT?

- **Pakiet**
  - Z adresu i portu (A, P_A).
  - Do adresu i portu (C, P_C).
  - NAT na podstawie krotki (A, P_A, C, P_C) wybiera port P_B.
  - Adres i port źródłowy pakietu podmienione na (B, P_B).

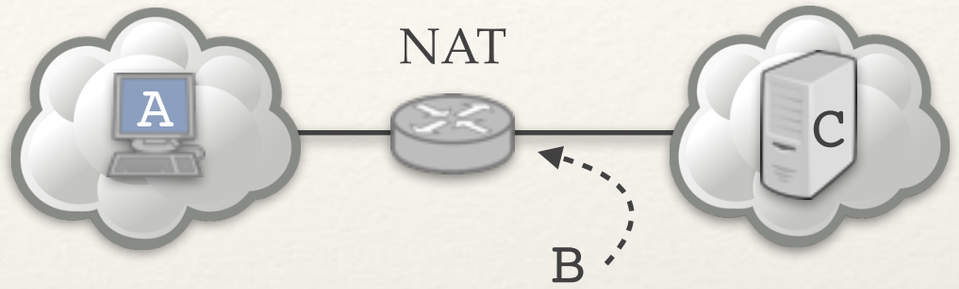


- **Tablica NAT:**
  - Przechowuje przez pewien czas przypisanie (A, P_A, C, P_C) → P_B.
  - **Dla kolejnych podobnych pakietów przypisanie będzie takie samo.**
  - Jeśli przychodzi pakiet **z Internetu** do (B, P_B) to jego adres i port docelowy zostanie podmieniony na (A, P_A).

NAT a P2P

Kiedyś:

- Komunikacja zawsze w modelu klient-serwer.
- Serwery nie są za routerami z NAT.
- Klienci mogą być za routerami z NAT

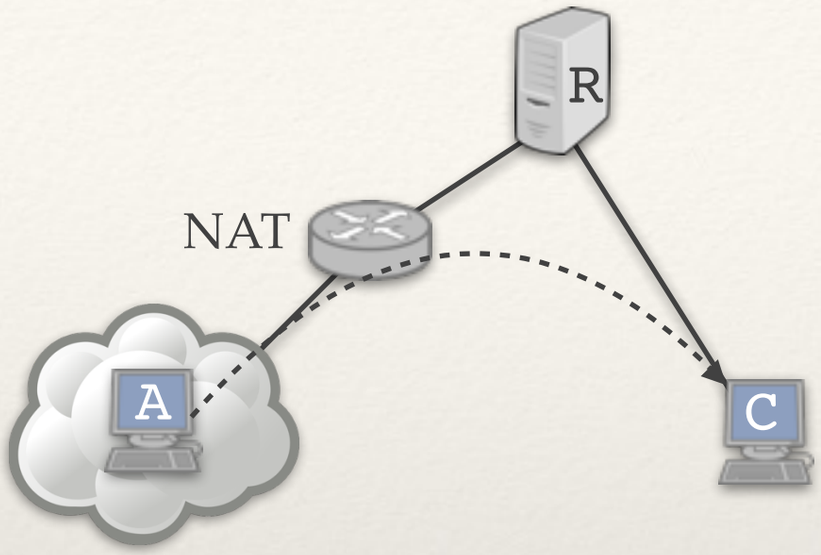


- Początkowa transmisja (np. TCP SYN) od klienta do serwera tworzy przypisanie $(A, P_A, C, P_C) \rightarrow P_B$, dzięki któremu pakiety z odpowiedziami serwera mogą wracać do klienta.

Obecnie:

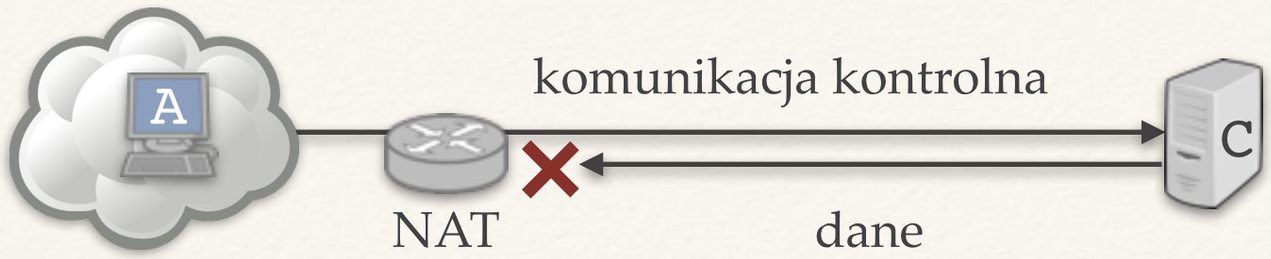
- Chcemy często przesyłać dane w modelu peer-to-peer (Bittorrent, Skype, …)
- Obie strony są często za NAT!
- Brak naturalnej możliwości zainicjowania połączenia.

Odwrócone połączenie



- C chce nawiązać połączenie z A, ale A jest za NAT.
- Jeśli oba utrzymują kontakt z R, to C może poprosić (przez R) komputer A o nawiązanie bezpośredniego połączenia z C.
- Stosowane np. w Skype.

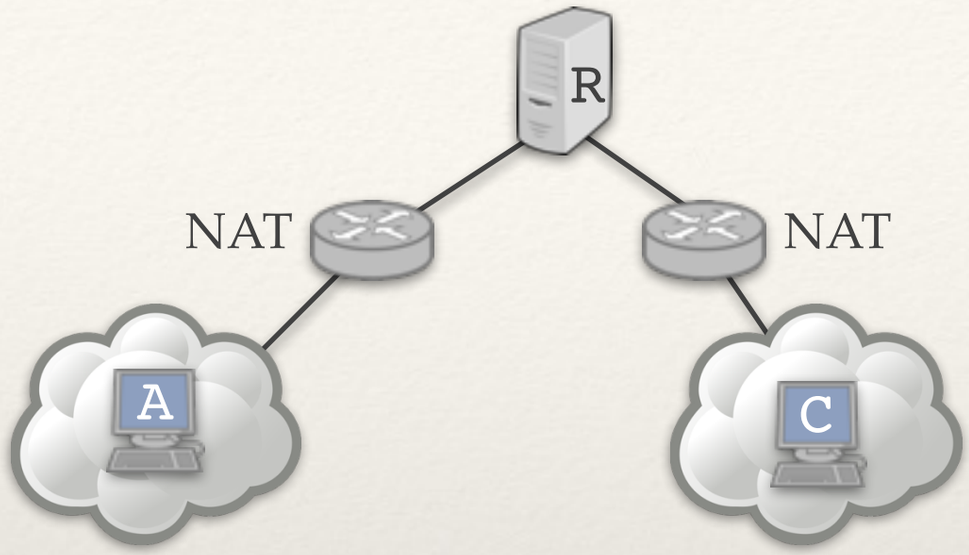
Odwrócone połączenie w protokole FTP



FTP: protokół przesyłania plików.

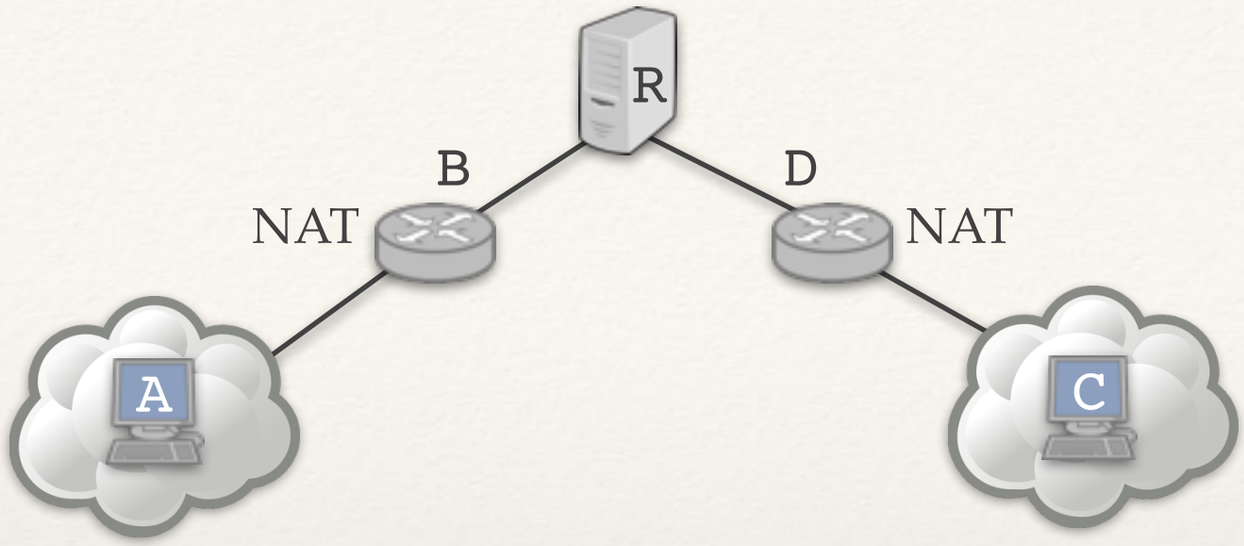
- Na początku klient A łączy się z serwerem C na porcie 21 (połączenie na komunikaty kontrolne).
- A wysyła polecenie „chcę pobrać plik i słucham na porcie X“
  → C łączy się z portem X klienta A i wysyła plik (odrębne połączenie TCP)
  → połączenie odrzucane przez NAT.
- Tryb pasywny FTP: A wysyła polecenie „chce pobrać plik w trybie pasywnym“
  → C zaczyna słuchać na porcie Y i wysyła komunikat „słucham na porcie Y“
  → A łączy się z portem Y serwera C i pobiera plik.

Przekaźniki



- Jeśli A i C utrzymują kontakt z R, to oba mogą nawiązać połączenie z R i R może przekazywać między nimi dane.
- Stosowane np. w Skype (jeśli wszystko inne zawiedzie).

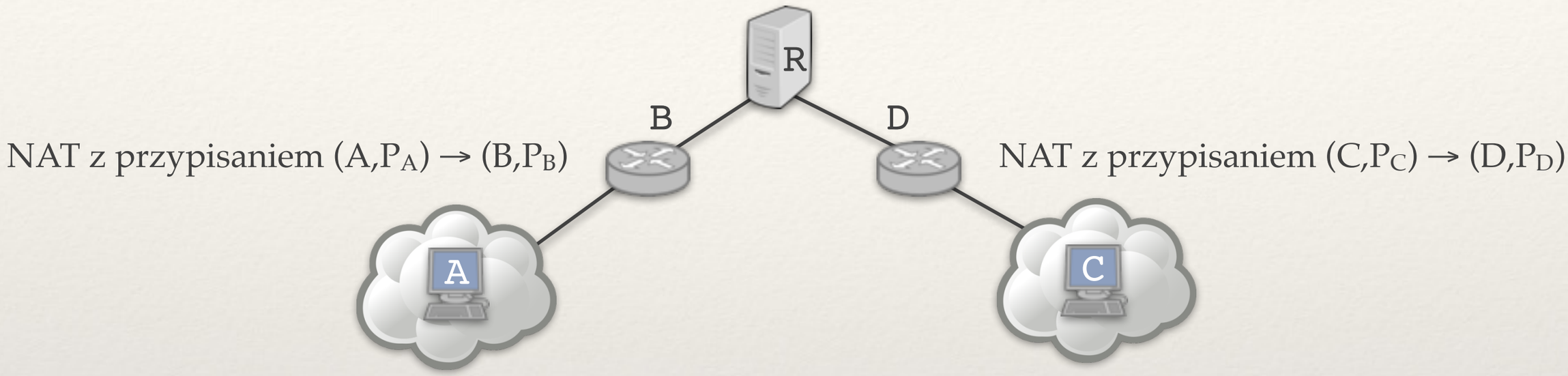
Przechodzenie przez NAT (1)



- A wysyła z portu P_A pakiet do R o treści „(A,P_A)“.
- Na routerze NAT zostaje utworzone przypisanie $(A,P_A) \rightarrow (B,P_B)$.
- R widzi pakiet o treści „(A,P_A)“ od (B, P_B), tj. poznaje przypisanie.
- W taki sam sposób R poznaje przypisanie $(C,P_C) \rightarrow (D,P_D)$.
- R odsyła poznane przypisania do A i C.

Przechodzenie przez NAT (2)

A łączy się z (D,P_D) → dane zostają przesłane do (C,P_C).

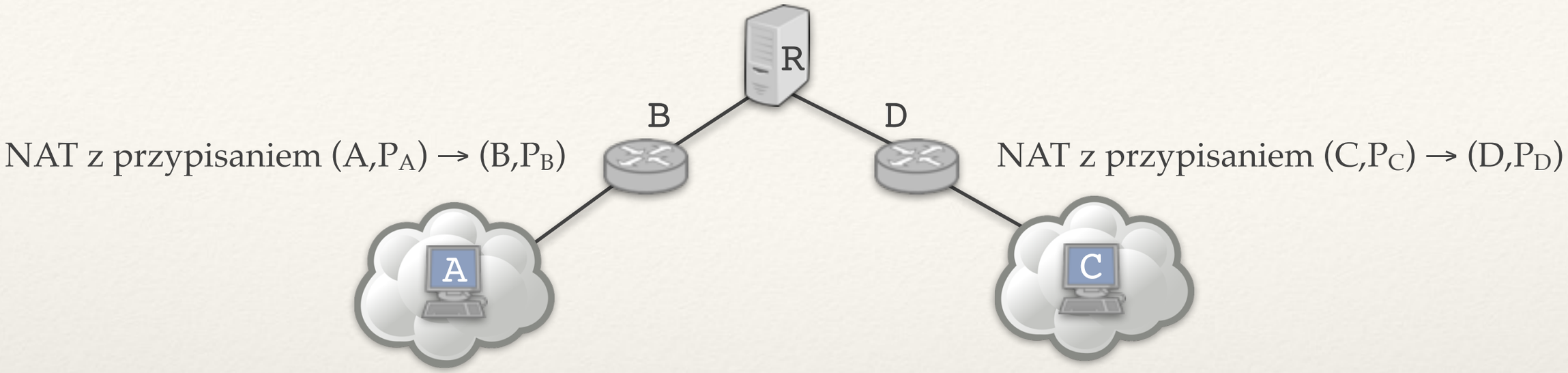


D poza przypisaniem (C,P_C) → (D,P_D) pamięta **listę odbiorców** pakietów, które wychodziły przez (D,P_D). D może wpuszczać:

zawiera (R,P_R)

- wszystkie pakiety (**pełny asymetryczny NAT**). ✓
- pakiety tylko od IP z listy (**ograniczony as. NAT**). ✗ od R
- pakiety tylko od portów z listy (**ogranicz. portowo as. NAT**). ✗ od (R,P_R)

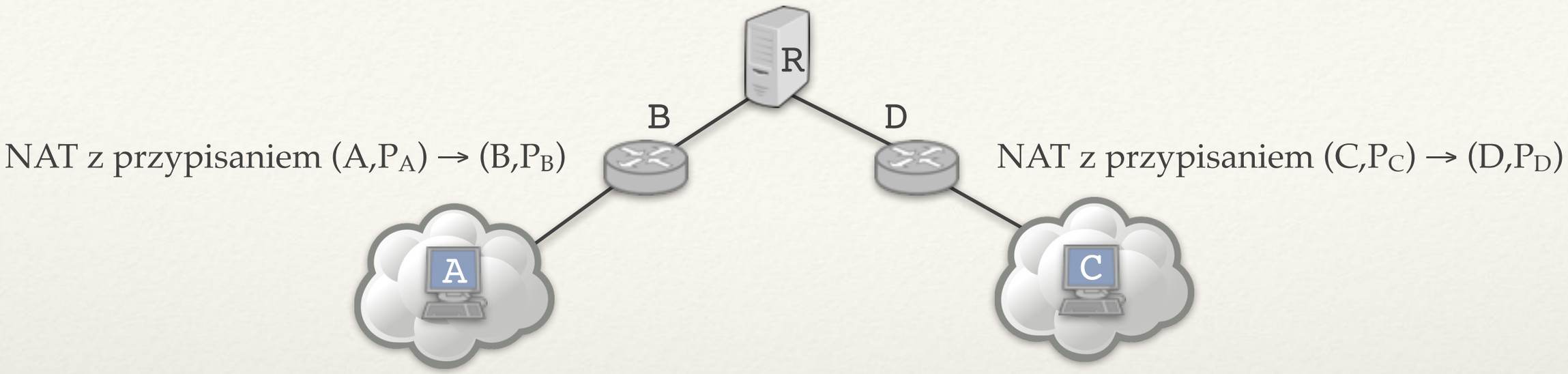
Wybijanie dziur (1)



Technika wybijania dziur (*hole punching*)

- (C,P_C) wysyła pakiet do (B,P_B). B odrzuca ten pakiet.
- Ale na routerze D lista odbiorców pakietów, które wychodziły przez (D,P_D) to teraz (R,P_R) i (B,P_B).
- Jeśli (A,P_A) wyśle teraz pakiet do (D,P_D), to D zobaczy to jako pakiet od (B,P_B) i przekaże do (C,P_C).
- Działa nawet dla asymetrycznych NAT ograniczonych portowo.

Wybijanie dziur (2)



- Milcząco założyliśmy, że jeśli (A,P_A) wysyłało pakiet do (R,P_R) i potem do (D,P_D), to w obu przypadkach B wybierze port P_B.
- **NAT asymetryczny:** P_B zależy tylko od adresu i portu nadawcy.
- **NAT symetryczny:** P_B zależy od adresu i portu nadawcy i odbiorcy. Nie wiadomo jak łączyć dwóch klientów za symetrycznymi NAT.

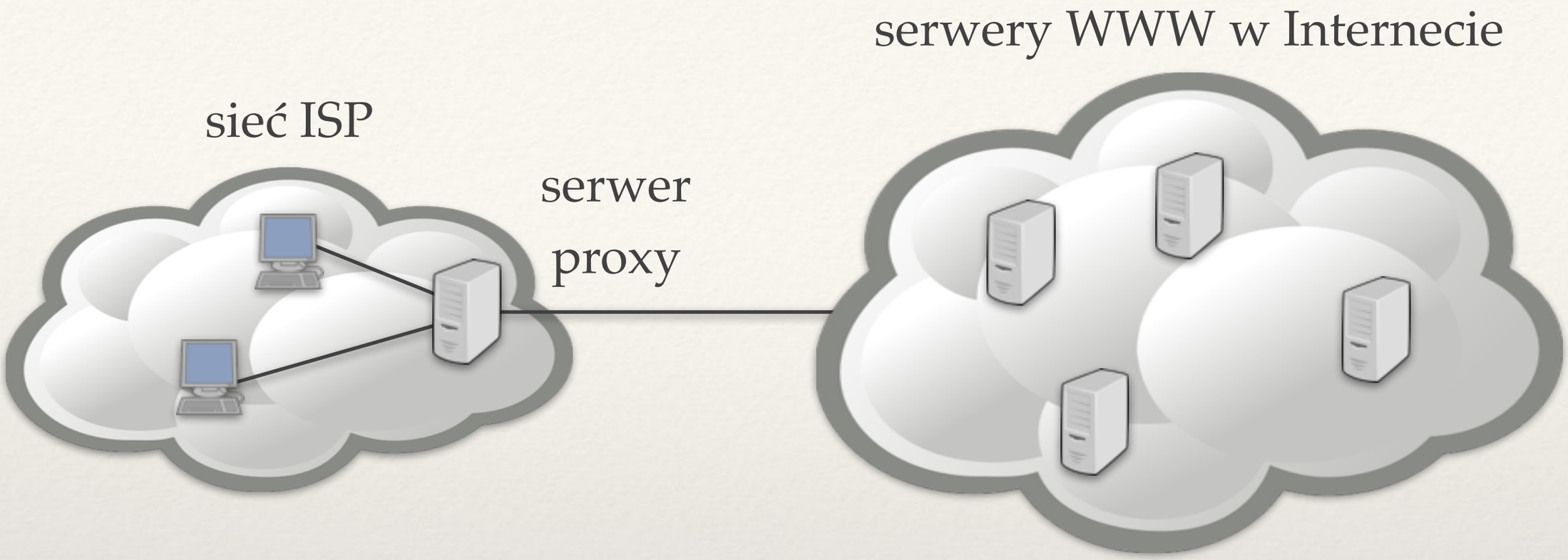
Wydajność HTTP

Poprawianie wydajności HTTP

- **Połączenia trwałe:**
  - Wiele żądań i odpowiedzi HTTP w jednym połączeniu TCP (standardowe zachowanie HTTP 1.1).

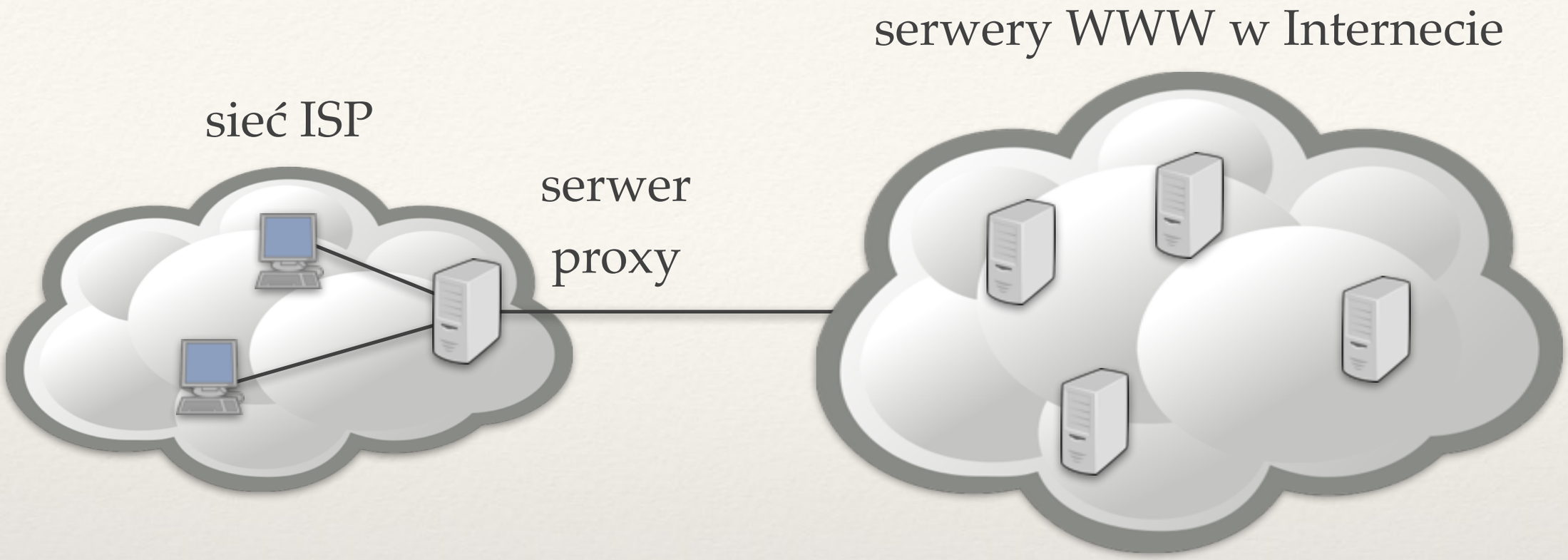
- **Pamięć podręczna w przeglądarce WWW:**
  - Zapytanie GET z polem `If-Modified-Since:`
  - Serwer może umieszczać w nagłówku odpowiedzi pola:
    - `Expires:` (do kiedy można trzymać dokument w pamięci podręcznej) → można całkowicie pominąć żądanie strony.
    - `Cache-Control: no-cache` (nigdy nie trzymaj w pamięci podręcznej)

Serwery proxy (1)



- Przeglądarka wysyła zapytanie HTTP do serwera proxy.
- Proxy w razie potrzeby łączy się z serwerem HTTP.
- Serwer proxy odpowiada używając stron przechowywanych w swojej pamięci podręcznej.
- W razie potrzeby przeglądarka może wymusić pominięcie proxy.

Serwery proxy (2)



Serwer proxy

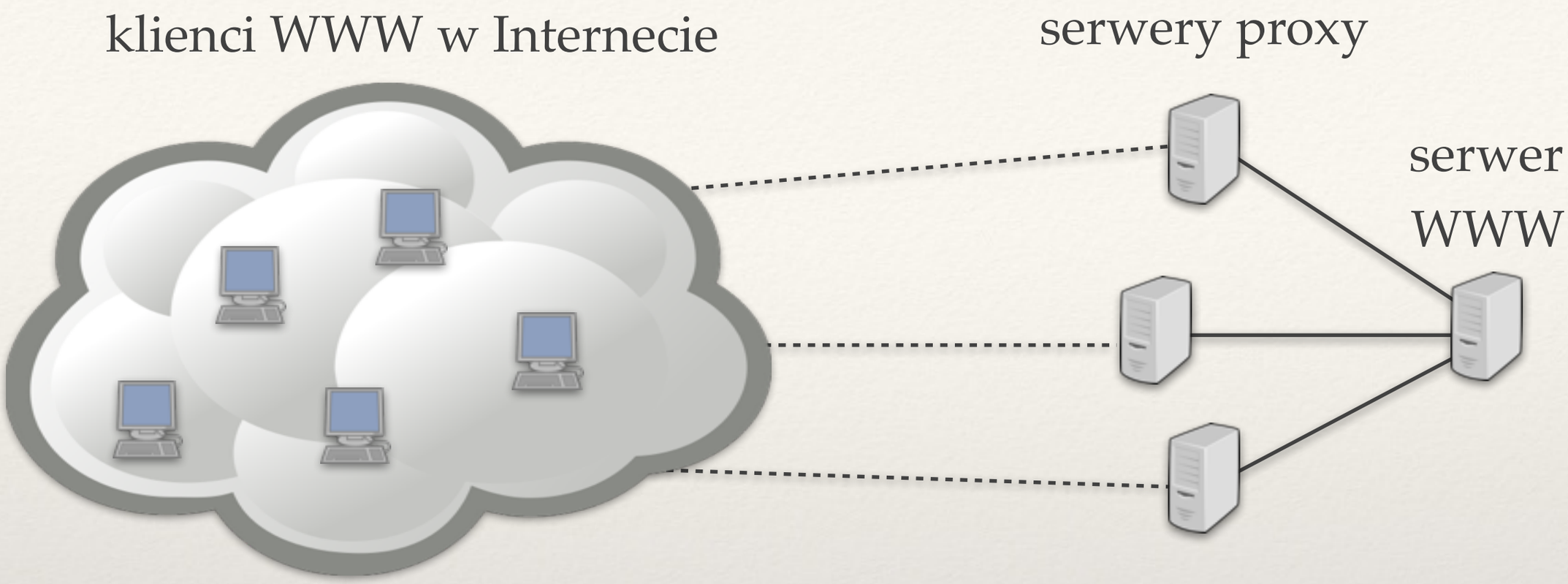
- Wpisywany w ustawieniach przeglądarki HTTP
- Czasem wymuszany przez ISP (integrowany z routerem obsługującym ruch z danej sieci).
- Korzyści: głównie dla ISP (ograniczenie ilości danych).

Anonimiowe serwery proxy

- **Serwer proxy dodaje do żądania HTTP dodatkowe pola.**
  - `X-Forwarded-For:` adres IP.
  - `Via:` adres IP proxy.

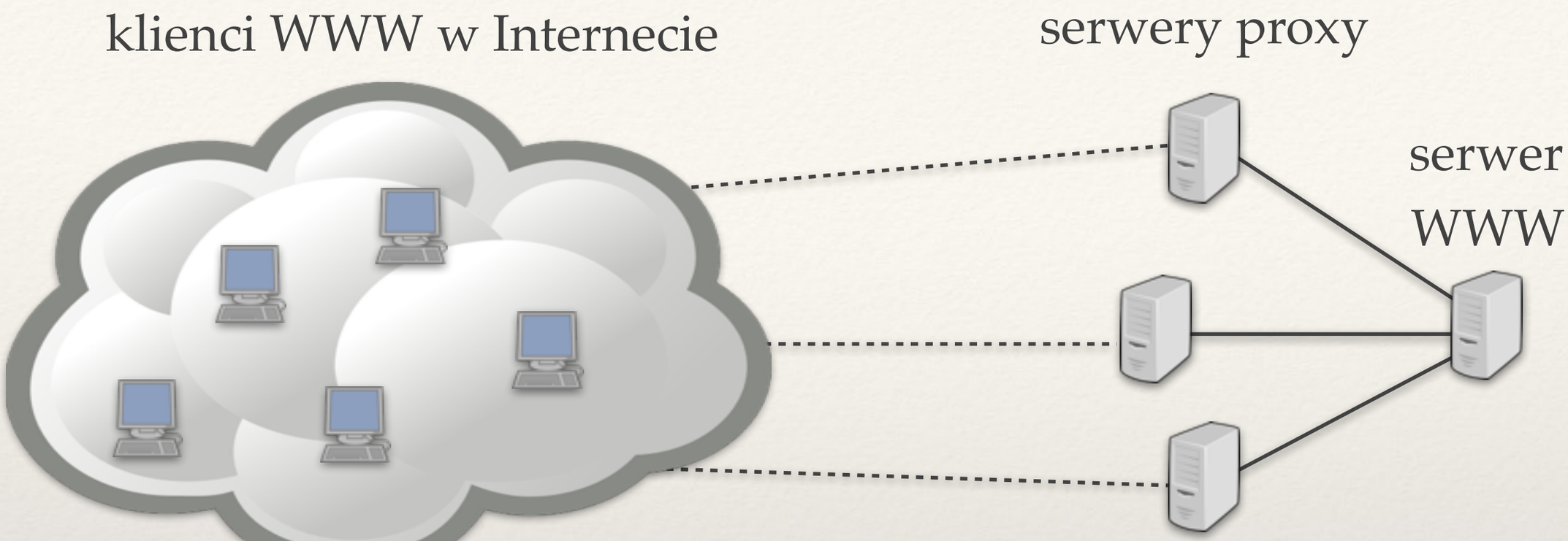
- **Anonimowe serwery proxy:**
  - Nie dodają takich nagłówków.
  - Zwykle płatne.

Odwrotne proxy (1)



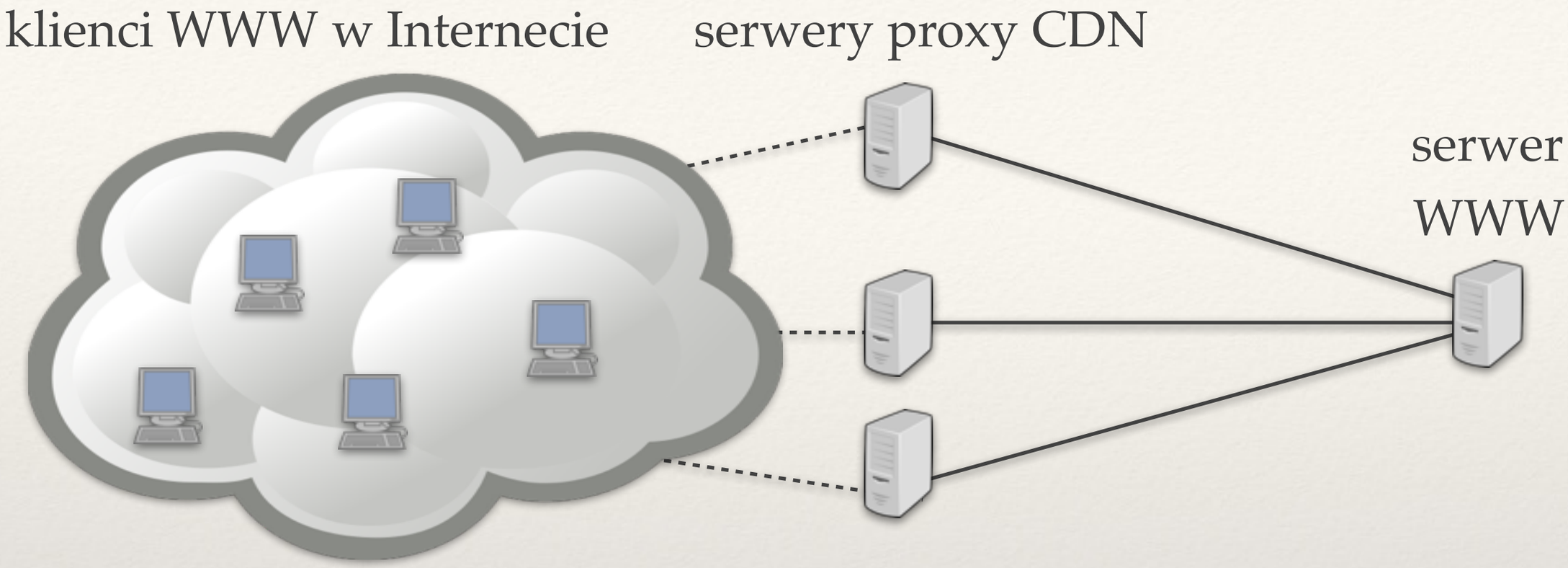
- Wykorzystywane przez dostawców treści.
- Zmniejszają obciążenie samego serwera WWW.
- Adresy IP serwerów proxy podawane zazwyczaj przez DNS jako adresy IP przy rozwiązywaniu nazwy serwera WWW.
  - Serwery DNS zazwyczaj zwracają listę adresów IP w losowej albo cyklicznej kolejności.

Odwrotne proxy (2)



- Zysk dla klienta i dostawcy treści.
- Ale wciąż duże opóźnienie w przesyłaniu pakietów pomiędzy klientami i serwerami proxy.
- Jak opłacalnie przysunąć serwery proxy do klientów?

CDN (Content Distribution Networks)



- Serwery proxy obsługiwane przez osobną organizację (obsługuje wiele serwerów WWW).
  - Akamai, Limelight, …
  - Setki tysięcy serwerów proxy.
- CDN utrzymuje również serwery DNS: umożliwiają wybieranie bliskiego serwera proxy.

Lektura dodatkowa

- Kurose & Ross: rozdział 2.
- Tanenbaum: rozdział 7.