



**AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE**

# **Sieci komputerowe**

**Protokoły warstwy aplikacji**

**dr inż. Andrzej Opaliński  
andrzej.opalinski@agh.edu.pl**

# Plan wykładu

- **Wprowadzenie – opis warstwy aplikacji**
- **Prezentacja protokołów**
  - Telnet
  - SSH
  
  - FTP
  - TFTP
  - SMB
  
  - SMTP
  - POP3
  - IMAP
  
  - HTTP
  - HTTPS
  - DNS
  - DHCP
  - LDAP
- **Podsumowanie**

# Funkcjonalności warstw niższych - przypomnienie

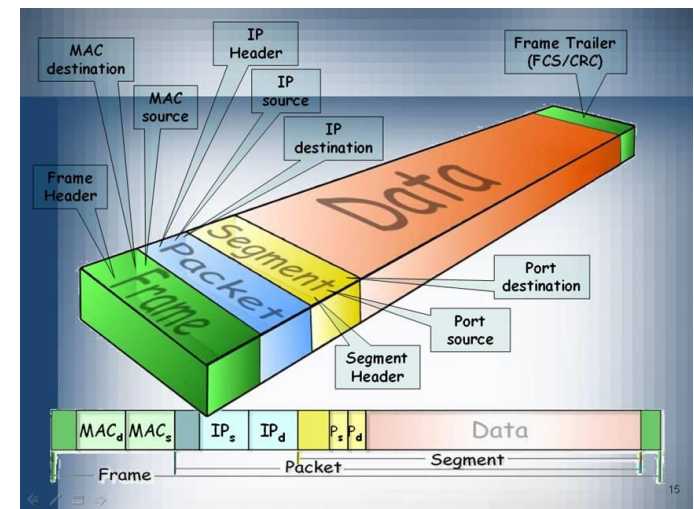
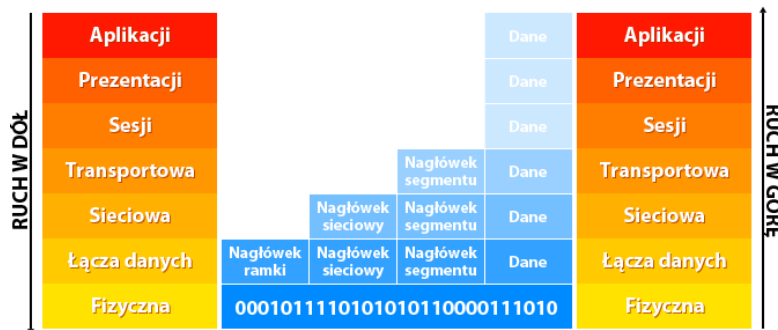
- **Warstwa prezentacji**

- Kodowanie i konwersja na postać odpowiednią dla architektury
  - Kompresja
  - Szyfrowanie
- w sposób umożliwiający dekompresję/deszyfrowanie u odbiorcy

- **Warstwa sesji**

- umożliwienie równoległego wykorzystania jednego interfejsu sieciowego przez wiele procesów (aplikacji)
- Obsługa wymiany informacji (inicjacja, restarty sesji po nieaktywności)

Warstwy w modelu odniesienia OSI



# Warstwa aplikacji – wprowadzenie

- **Warstwa aplikacji (ang. process layer)**
  - Specyfikacja interfejsu, który wykorzystują aplikacje (procesy) do przesyłania danych do sieci (poprzez warstwy niższe)
  - „świadczy usługi końcowe dla aplikacji”
  - Obejmuje zestaw protokołów wykorzystywanych przez aplikacje
  - Różnice ISO/OSI a TCP/IP

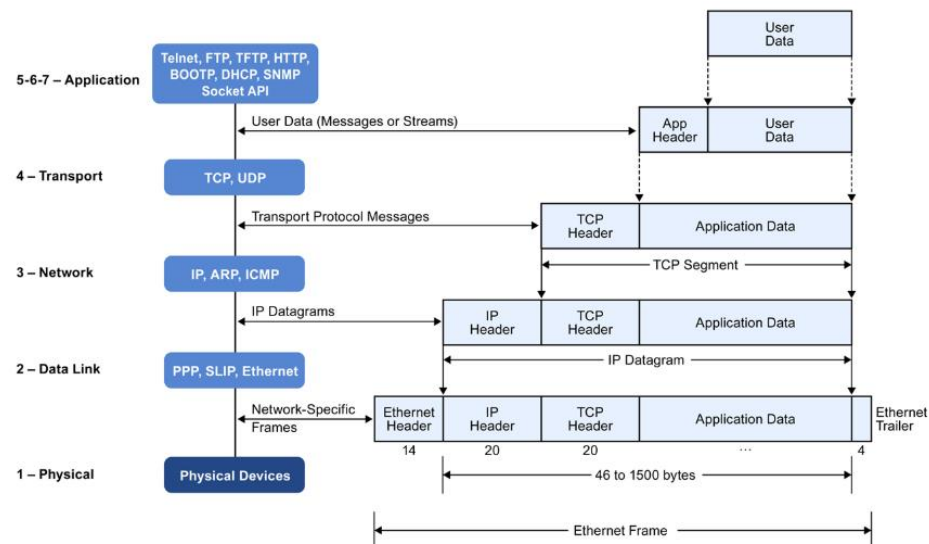
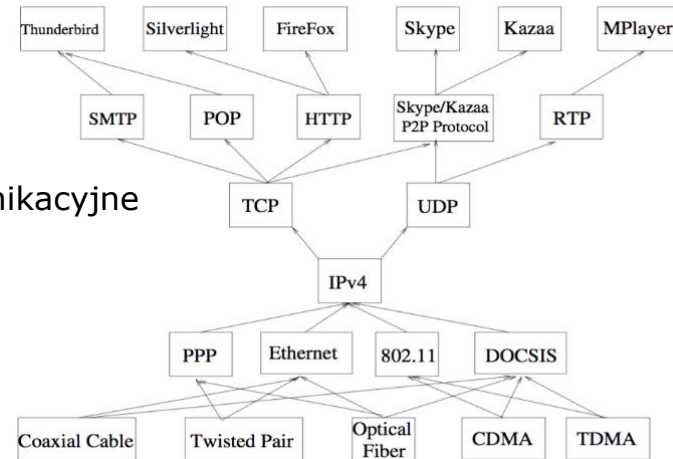
OSI	TCP/IP
Application	Application
Presentation	
Session	Transport
Transport	
Network	Network
Data link	Physical
Physical	

- Definicja

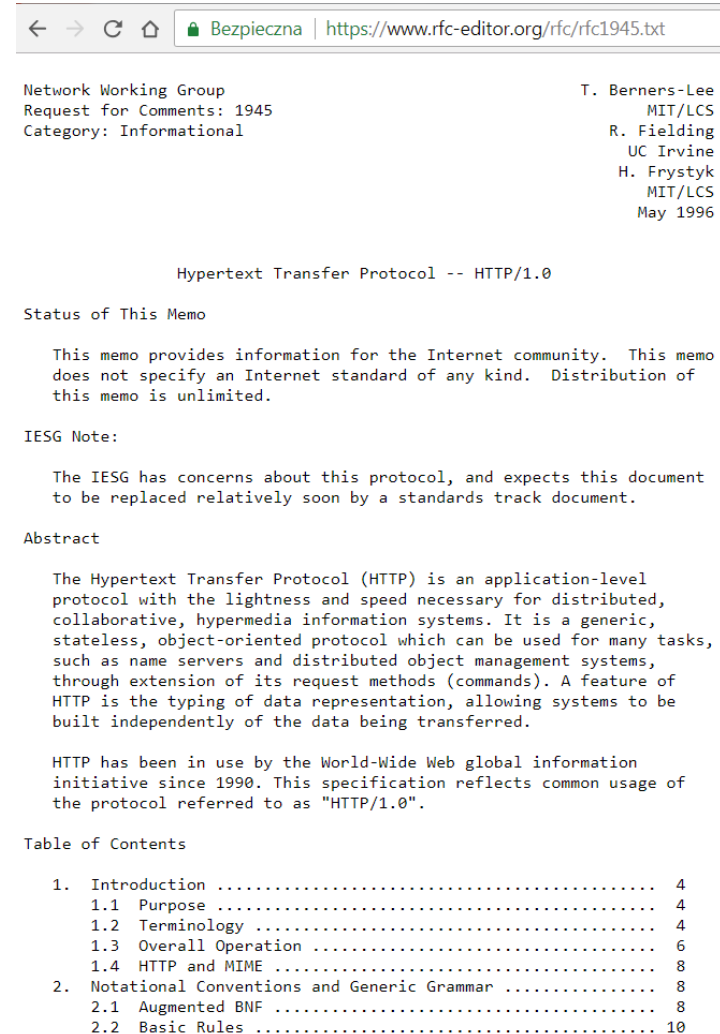
- zbiór reguł i kroków postępowania, wykonywanych automatycznie przez urządzenia komunikacyjne w celu nawiązania łączności i wymiany danych

- Elementy składowe protokołów

- Procedura początkowa (ustalenie parametrów połączenia: adresy, szybkość, rozmiar buforów itp.)
- Procedura transmisji danych (format ramki)
- Procedura analizy poprawności przekazu (sumy kontrolne)
- Procedury retransmisji
- Procedury zakończenia połączenia
- Procedury (de)fragmentacji danych



- **Request For Comments (RFC)**
  - Definiują szczegóły protokołów w ramach modelu OSI i TCP/IP
  - Utrzymywane i obsługiwane przez IETF
- **Internet Engineering Task Force**  
<https://www.ietf.org/standards/rfcs/>
- Wyszukiwarka dokumentów
- Dostępne w formatach
  - Tekstowym
  - HTML
  - PDF



← → ↻ 🏠 Bezpieczna | <https://www.rfc-editor.org/rfc/rfc1945.txt>

Network Working Group  
Request for Comments: 1945  
Category: Informational

T. Berners-Lee  
MIT/LCS  
R. Fielding  
UC Irvine  
H. Frystyk  
MIT/LCS  
May 1996

Hypertext Transfer Protocol -- HTTP/1.0

Status of This Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

IESG Note:

The IESG has concerns about this protocol, and expects this document to be replaced relatively soon by a standards track document.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing of data representation, allowing systems to be built independently of the data being transferred.

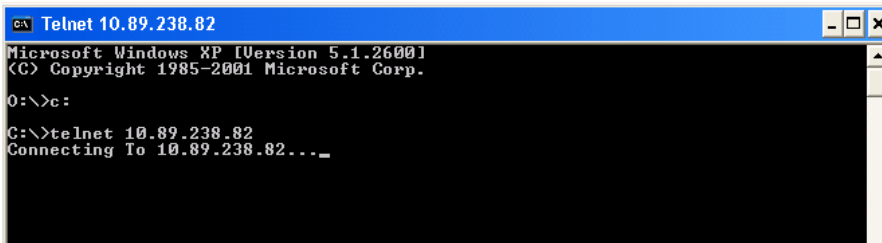
HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification reflects common usage of the protocol referred to as "HTTP/1.0".

Table of Contents

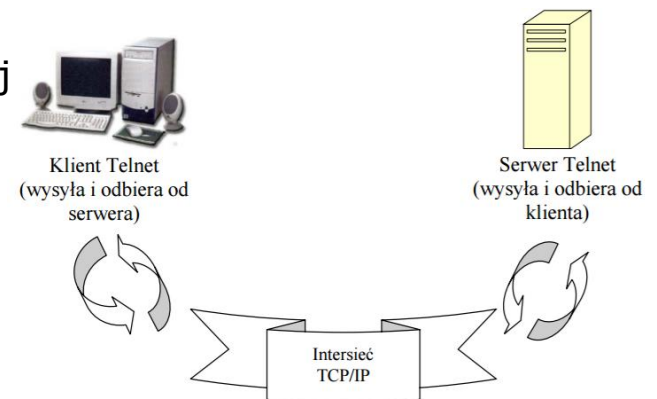
1. Introduction .....	4
1.1 Purpose .....	4
1.2 Terminology .....	4
1.3 Overall Operation .....	6
1.4 HTTP and MIME .....	8
2. Notational Conventions and Generic Grammar .....	8
2.1 Augmented BNF .....	8
2.2 Basic Rules .....	10

- **Protokół do obsługi zdalnego terminala w architekturze klient-serwer**

- Wykorzystuje protokół TCP i port 23, RFC.854
- Obsługuje jedynie terminale alfanumeryczne w trybie poleceń (brak obsługi myszy, GUI)
- Działa w trybie klient-serwer
  - Na komputerze użytkownika – klient telnet
  - Na serwerze – telnetd (Unix), Telnet server (Windows)
- Wymaga posiadania konta na serwerze
  - Osobistego (user/password)
  - Otwartego (ograniczony dostęp do aplikacji)
- Wykorzystywany do konfiguracji zdalnej urządzeń aktywnych w sieci
  - Przełączniki
  - Routery
- Wada – przesyłanie hasła w formie niezaszyfrowanej
- Rozwiązanie – zastępowany przez SSH



```
cmd Telnet 10.89.238.82
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
O:\>c:
C:\>telnet 10.89.238.82
Connecting To 10.89.238.82....
```





AGH

# SSH

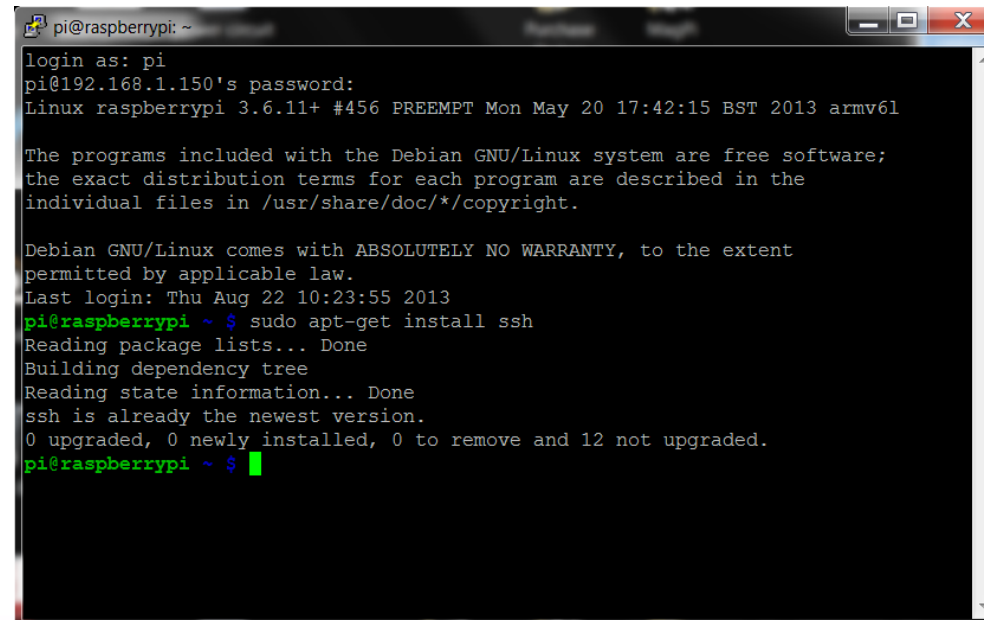
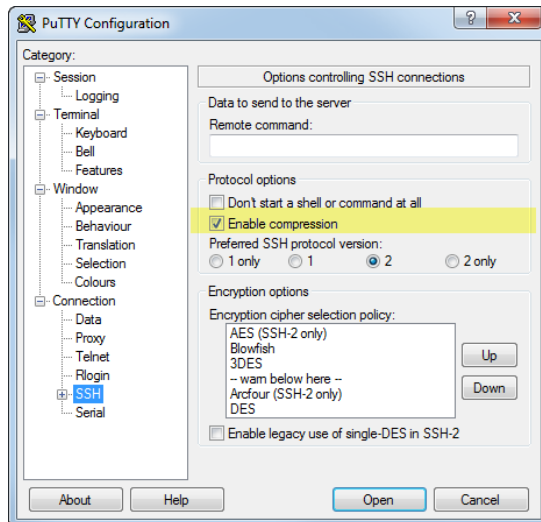
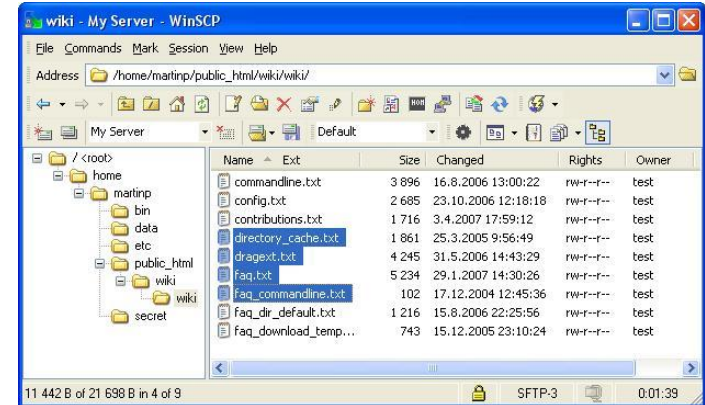
- **SSH (ang. secure shell)**
  - Standard protokołów komunikacyjnych w architekturze klient-serwer
    - Terminalowe (ssh)
    - Przesyłania plików (scp, sftp)
    - Zdalnej kontroli zasobów (rsync over ssh)
    - Tunelowanie (portów, sesji okien X11)
  - Następcą protokołu telnet (zastępuje także rlogin, rsh, rexec)
  - Transfer wszystkich danych jest zaszyfrowany
  - Działa standardowo na porcie 22
- Dwie wersje protokołu
  - SSHv1
    - Stała lista metod szyfrowania
    - Dwa sposoby uwierzytelniania (klucz RSA oraz hasło)
  - SSHv2
    - Nowa metoda wymiany klucza (Diffie-Hellman)
    - Sprawdzanie spójności transmisji (kody autentyfikacji)
    - Dowlone sposoby szyfrowania
    - Cztery sposoby uwierzytelniania (hasło, klucze (RSA,DSA), protokół Kerberos)
- **Najczęstsze metody szyfrowania: AES, Blowfish, DES**  
(3DES, BLOWFISH, TWOFISH256, TWOFISH, TWOFISH192, TWOFISH128, AES256, AES192, AES128, SERPENT256, SERPENT192, SERPENT128, ARCFOUR, IDEA, CAST128)





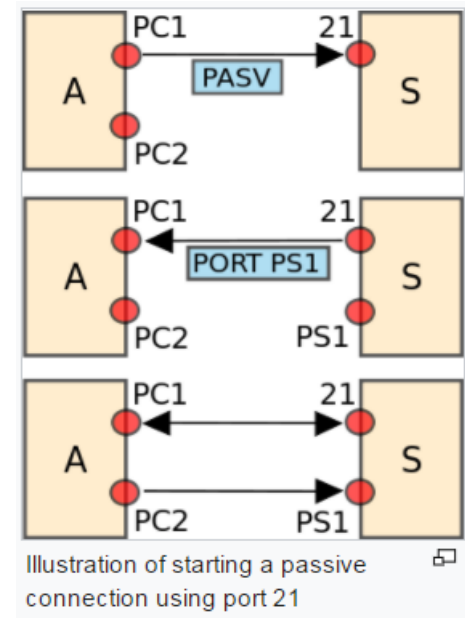
# AGH SSH – c.d.

- Najważniejsze implementacje
  - Zamknięta: ssh.com
  - Otwarta: OpenSSH
- Klienci
  - Aplikacje użytkownika łączące się z serwerem SSH (ssh daemon)
  - PuTTY
  - OpenSSH
  - WinSCP



- **FTP (ang. File Transfer Protocol) – protokół transferu plików**

- Protokół typu klient-serwer wykorzystujący protokół TCP, RFC 959
- Wykorzystuje dwa połączenia TCP
  - Połączenie kontrolne – przesyłanie poleceń
  - Połączenie do transmisji danych
- Może działać w dwóch trybach
  - Tryb aktywny
    - Port 21 – polecenia (zestawiane przez klienta)
    - Port 20 – transfer danych (zestawiane przez serwer)
  - Tryb pasywny
    - Port 21 – polecenia (zestawiane przez klienta)
    - Port powyżej 1024 – transfer danych (zestawiane przez klienta)
- Tryby dostępu
  - Anonimowy – bez hasła uwierzytelniającego
  - Autoryzowany – w oparciu o login i hasło (niezaszyfrowane)



- Reprezentacje danych
  - ASCII – wykorzystywany do transmisji danych tekstowych, konwertowany do 8 bitowego kodu ASCII
  - Image/Binary – wysyłanie pliku jako strumienia bajtów,
  - EBCDIC – wysyłanie tekstu (plain text) w dla systemu kodowania EBCDIC
  - Local mode – tryb dla dwóch komputerów w identycznej konfiguracji we własnym trybie kodowania bez konieczności konwersji na ASCII
- Tryby transferu danych
  - Transmisja strumieniowa - bez dodatkowego przetwarzania, pozostałe funkcjonalności przerzucone na protokół TCP
  - Transmisja blokowa – FTP dzieli dane na bloki (nagłówek, długość, dane) i takie segmenty przekazuje do przesłania przez TCP
  - Transmisja skompresowana

Descrptr	Byte count	
code= 16		= 6

Marker	Marker	Marker
8 bits	8 bits	8 bits

Marker	Marker	Marker
8 bits	8 bits	8 bits

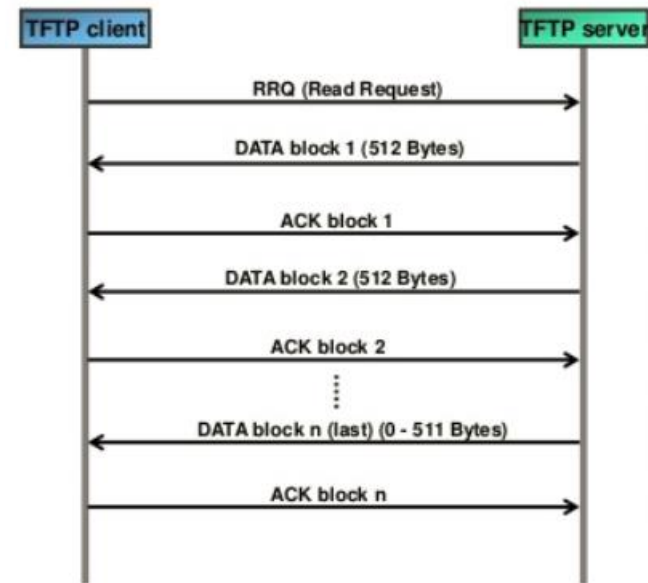


AGH

# TFTP

- **Trivial File Transfer Protocol**
- **Mniej funkcjonalności w porównaniu do FTP**
  - Bez wyświetlania katalogów
  - Bez uwierzytelniania użytkowników
- **Podstawowa funkcjonalność**
  - Odczytywanie/wysyłanie plików z/na komputer zdalny
  - Pliki transmitowane w blokach po 512 bajtów (blok krótszy niż 512 bajtów oznacza koniec pliku)
  - Implementowany na protokole UDP
    - Nasłuchiwanie na porcie 69
    - Transmisja danych na innych portach
- **Gwarancja niezawodności transmisji**
  - Każdy pakiet wymaga odesłania pakietu potwierdzającego
  - Retransmisja ostatniego pakietu po „przeterminowaniu” czasu oczekiwania
- **Trzy tryby przesyłania**
  - Netascii – 7-bitowy kod ASCII + specyfikacja protokołu Telnet (RFC 854)
  - Oktet – przesyłanie informacji bit po bicie. Podobny do trybu binarnego protokołu FTP
  - Poczta – niewykorzystywany

## TFTP read request (RRQ):





# AGH TFTP - kontynuacja

## • Pięć typów pakietów

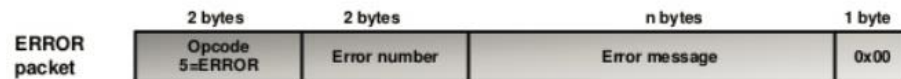
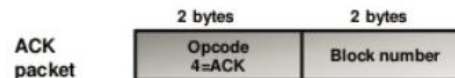
- RRQ – żądanie odczytu
- WRQ – żądanie zapisu
- DATA – dane
- ACK – potwierdzenie
- ERROR – błąd

## • Pola pakietów RRQ i WRQ

- Opcode – kod operacji.
- Filename – nazwa pliku, który ma zostać przesłany
- Filename terminator – 8-bitowe pole (0) - koniec pola Filename.
- Mode – pole zawierające tekst: "netascii", "octet" lub "mail"
- Mode terminator – 8-bitowe pole zawiera wartość zero wskazujące koniec pola Mode.

## • Kody błędów

- 0 Niezdefiniowany. Sprawdź komunikat o błędzie (jeżeli jest).
- 1 Nie znaleziono pliku.
- 2 Naruszenie dostępu.
- 3 Dysk pełen lub przekroczona alokacja.
- 4 Zabroniona operacja TFTP.
- 5 Nieznana tożsamość przesyłu.
- 6 Plik już istnieje.
- 7 Nie ma takiego użytkownika.



cecha	FTP	TFTP
Autentyfikacja	W oparciu o login i hasło	Brak autentyfikacji
Połączenie	Oparte na TCP. Retransmisja w oparciu o mechanizmy zapewnione w TCP	Oparte na UDP. Błędy (utracone pakiety, sumy kontrolne) oparte o mechanizmy TFTP.
Algorytm protokołu	W oparciu o mechanizmy zapewniana w TCP. (kontrola przepływu (okno przesuwne, bufor) i przeciążeń)	Oparty na potwierdzeniach przesyłanych pakietów (pojedynczych, ograniczona przepustowość).
Złożoność	Nieco bardziej skomplikowany od TFTP. Czasami niezalecany do bootloaderów przechowywanych w pamięci EEPROM.	Bardzo prosty, z bardzo małym narzutem (oparty o równie prosty UDP). Może być wykorzystywany w bootloaderach.
Kanały transmisji danych	Dwa oddzielne kanały transmisji. Jeden do kontroli drugi do przesyłania danych w oparciu o dwa oddzielne połączenia TCP	Dane i informacje kontrolne przesyłane w oparciu o jedno połączenie TCP.

- **Simple Mail Transfer Protocol**

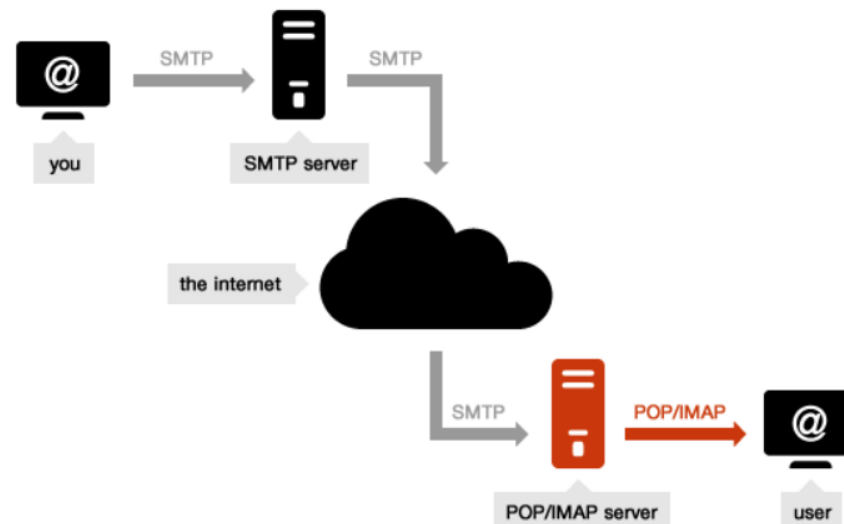
- Protokół komunikacyjny opisujący sposób przekazywania poczty w Internecie
- Definicja standardu w RFC 821 i RFC 532
- Demon SMTP działa zwykle na porcie 25 protokołu TCP (czasami na 587)
- Bezpieczne połączenie w warstwie transportowej w oparciu o protokół TLS -> SMTPS

- **Funkcjonalności**

- Wykorzystywany przez **serwery pocztowe** do **wysyłania i odbierania** wiadomości
- Pocztove **aplikacje klienckie** (outlook, thunderbird, ...) używają go zwykle **jedynie do wysyłania** poczty

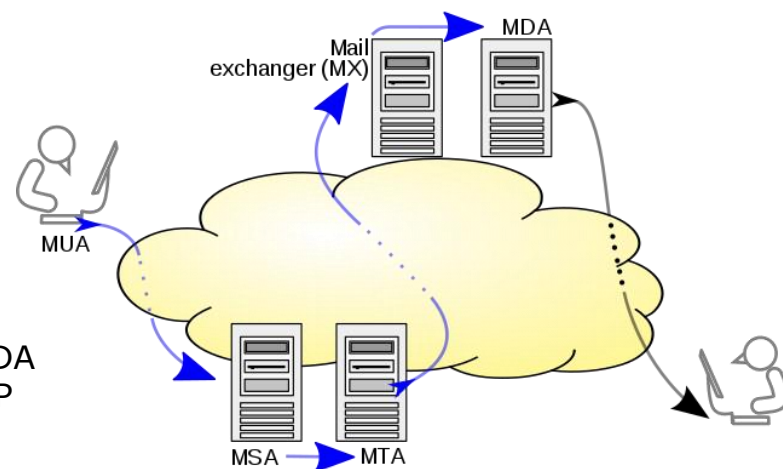
- **Serwery pocztowe**

- Unix/Linux
  - Exim – free/GPL2 (57% of mail servers)
  - Postfix – open source (34% of mail serv)
  - SendMail – free & proprietary
- Dla Windows
  - hMailServer – aGPL
  - IceWarp (także na Linuxach) – prop.



## • Proces przetwarzania wiadomości e-mail

- Mail user agent (mail client) – łączy się z MSA na porcie 587 lub 25
- Mail submission agent – przesyła dane do MTA
- Mail transfer agent (może być zintegrowany z MSA) - w oparciu o rekord MX z systemu DNS wyszukuje MX dla domeny docelowej
- Mail exchanger – odbiera wiadomości dla danej domeny (jako punkt ostateczny, lub pośrednik) i kieruje do docelowego MDA
- Mail delivery agent – przechowuje wiadomość i udostępnia ją do pobrania dla użytkownika
- Mail user agent (mail client) – autentyfikuje się w MDA i pobiera wiadomości poprzez protokoły IMAP lub POP







## • Podstawowe komendy protokołu SMTP

- HELO – C->S, identyfikacja klienta na serwerze i inicjalizacja komunikacji
- EHLO – jak HELO, gdy klient chce użyć ESMTP
- MAIL – specyfikacja adresu nadawcy
- RCPT – specyfikacja adresu odbiorcy
- DATA – początek treści wiadomości (na zakończenie kropka w nowej linii)
- RSET – przerwanie wysyłania maila i wyczyszczenie informacji o nadawcy/odbiorcach
- NOOP – żądanie odpowiedzi OK (PING)
- QUIT – C->S, żądanie zakończenia połączenia
- VRFY – żądanie zweryfikowania istnienia klienta na danym serwerze

## • ESMTP – Extended SMTP

- Rozszerzenie protokołu SMTP
- Dodatkowe komendy SEND, SOML, SAML, EXPN, HELP, TURN
- Rozszerzenie istniejących (MAIL, RCTP)

## • Przykładowa komunikacja

- W domenie example.com
- Pomiędzy bob@
- Do alice@, CC dla theboss@

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.com
S: 250 smtp.example.com, I am glad to meet you
C: MAIL FROM:<bob@example.com>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.com>
C: To: Alice Example <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```



AGH

# POP3

- **Post Office Protocol**

- Protokół klientów pocztowych do odbierania wiadomości email z serwerów pocztowych
- Aktualna wersja trzecia – POP3
- Aktualnie częściowo zastąpiony przez protokół IMAP

- **Klasyczny tryb działania**

- Połączenie klienta z serwerem
- Pobranie wszystkich wiadomości z serwera do klienta
- Zapisanie wiadomości w systemie klienta
- Usunięcie wiadomości z zasobów serwera

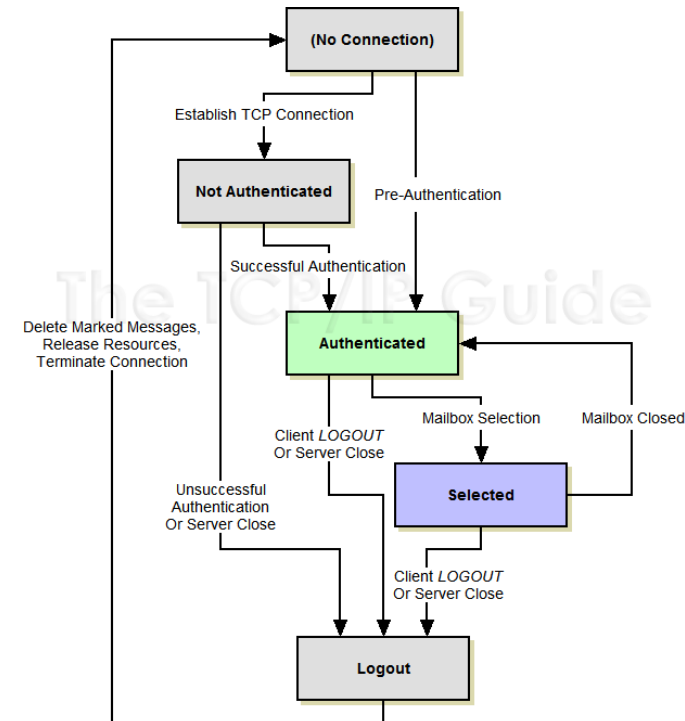
- **Cechy dodatkowe**

- Opcja umożliwia zachowanie wiadomości na serwerze
- Stosunkowo wąski zakres funkcjonalności związanych z zarządzaniem skrzynką pocztową
- Serwer nasłuchuje na porcie 110 TCP
- POP3 wymaga komunikacji szyfrowanej po inicjalizacji połączenia, w oparciu o:
  - komendę STLS (STARTTLS)
  - Mechanizm POP3S – w oparciu o połączenie TLS lub SSL na porcie 995

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK mrose's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C: <close connection>
S: <wait for next connection>
```

- **Internet Message Access Protocol**

- Protokół klientów pocztowych służący do pobierania wiadomości email z serwera pocztowego
- W założeniu zaprojektowany do obsługi skrzynki pocztowej przez **wielu klientów**
- standardowo pozostawia wiadomości na serwerze
- Nadbudowany nad połączeniem TCP/IP
- Serwer IMAP nasłuchuje standardowo na porcie 143
- IMAPS (rozszerzony o SSL) wykorzystuje port 993





AGH

IMAP

- **Przewagi nad protokołem POP**

- Możliwość równoległego podłączenia wielu klientów do jednej skrzynki wraz z zapewnieniem ich synchronizacji
- Dostęp do pojedynczych elementów wiadomości email typu MIME (Multipurpose Internet Mail Extensions) bez konieczności pobierania całości wiadomości
- Monitorowanie stanu wiadomości poprzez flagi (przeczytana, z odpowiedzią, skasowana, ...)
- Możliwość tworzenia, zmiany nazw i usuwania folderów (mailbox) oraz wykorzystywania folderów publicznych
- Wyszukiwanie wiadomości po stronie serwera bez konieczności ich pobierania

- **Przykład komunikacji**

```
C: <open connection>
S: * OK IMAP4rev1 Service Ready
C: a001 login mrc secret
S: a001 OK LOGIN completed
C: a002 select inbox
S: * 18 EXISTS
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * 2 RECENT
S: * OK [UNSEEN 17] Message 17 is the first unseen message
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: a002 OK [READ-WRITE] SELECT completed
C: a003 fetch 12 full
S: * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
"IMAP4rev1 WG mtg summary and minutes"
(("Terry Gray" NIL "gray" "cac.washington.edu"))
(("Terry Gray" NIL "gray" "cac.washington.edu"))
(("Terry Gray" NIL "gray" "cac.washington.edu"))
((NIL NIL "imap" "cac.washington.edu"))
((NIL NIL "minutes" "CNRI.Reston.VA.US")
("John Klensin" NIL "KLENSIN" "MIT.EDU")) NIL NIL
"<B27397-0100000@cac.washington.edu>")
BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028
92))
S: a003 OK FETCH completed
C: a004 fetch 12 body[header]
S: * 12 FETCH (BODY[HEADER] {342}
S: Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S: From: Terry Gray <gray@cac.washington.edu>
S: Subject: IMAP4rev1 WG mtg summary and minutes
S: To: imap@cac.washington.edu
S: cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>
S: Message-Id: <B27397-0100000@cac.washington.edu>
S: MIME-Version: 1.0
S: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S: )
S: a004 OK FETCH completed
C: a005 store 12 +flags \deleted
S: * 12 FETCH (FLAGS (\Seen \Deleted))
S: a005 OK +FLAGS completed
C: a006 logout
S: * BYE IMAP4rev1 server terminating connection
S: a006 OK LOGOUT completed
```



AGH

# HTTP

- **Hypertext Transfer Protocol**

- „protokół przesyłania dokumentów hipertekstowych w sieci WWW”
- Dokumenty hipertekstowe – „ustrukturalizowany tekst z hiperlinkami pomiędzy węzłami zawierającymi tekst”
- Działa jako protokół żądania(klient) - odpowiedzi(serwer)
- Jest protokołem bezstanowym – serwer nie monitoruje zmian/statusów/historii swojej komunikacji z klientem (server side sessions: cookies, hidden forms vars)

- **Sesja HTTP**

- Sekwencja transakcji żądania-odpowiedzi (request-response)
- Zwykle w oparciu o protokół TCP i port 80
- Serwer nasłuchuje na porcie i odpowiada na żądania klienta (Statusem oraz treścią wiadomości, najczęściej zasobem/contentem)

- **Metody HTTP**

- Określają akcje jakie mogą zostać wykonane na zasobach
- Zasobami mogą być pliki lub dynamicznie wygenerowany content (przez skrypty)
- Wersja HTTP 1.0 (1996 rok) GET, POST, HEAD
- Wersja HTTP 1.1 (1997 rok) dodała: OPTIONS, PUT, DELETE, TRACE, CONNECT
- HTTP/2 – specyfikacja opublikowana w 2015r
  - metody jak w HTTP 1.1
  - Nowe sposoby tworzenia ramek i transportu między klientem a serwerem (kompresja nagłówek, HTTP server push, kolejgowanie zapytań)

- **Metody HTTP**

- GET – pobranie z serwera do klienta zasobu wskazanego przez URI (Uniform Resource Identifier)
  - POST – żądanie odebrania przez serwer danych (np. formularza, komentarza, rekordu bazy danych) przesłanych od klienta w ramach zapytania identyfikowanych poprzez URI. (nie jest idempotentne)
  - HEAD – pobranie informacji o danym zasobie z serwera (ale bez jego zawartości)
  - OPTIONS - informacje o metodach obsługiwanych w ramach danego zasobu
  - PUT – żądanie odebrania i utworzenia/zaktualizowania danych wysłanych od klienta do serwera (jest idempotentne)
  - DELETE – żądanie usunięcia zasobu z serwera
  - TRACE – odesłanie przez serwer odebranego zapytania w celu przetestowania ewentualnych narzutów/modyfikacji serwerów pośredniczących
  - CONNECT – konwersja żądania na tunel TCP/IP w celu stosowania komunikacji szyfrowanej
  - PATCH – częściowa modyfikacja zasobu
- Idempotencja – wiele identycznych wywołań takiego samego żądania ma identyczny skutek co pojedyncze wywołanie tego żądania. (dotyczy stanu serwera po obsłudze żądania)



- **Kody statusu**

- Pierwsza linia odpowiedzi zawiera liczbowy status oraz tekstowe rozszerzenie
- 1XX – Informacyjny
- 2XX – Powodzenie
- 3XX – Przekierowanie
- 4XX – Błąd klienta
- 5XX – Błąd serwera

**Client error responses**

**400 Bad Request**  
The server could not understand the request due to invalid syntax.

**401 Unauthorized**  
Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested response.

**402 Payment Required**   
This response code is reserved for future use. The initial aim for creating this code was using it for digital payment systems, however this status code is used very rarely and no standard convention exists.

**403 Forbidden**  
The client does not have access rights to the content, that is, it is unauthorized, so the server is refusing to give the requested resource. Unlike 401, the client's identity is known to the server.

**404 Not Found**  
The server can not find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 to hide the existence of a resource from an unauthorized client. This response code is probably the most famous one due to its frequent occurrence on the web.

**Successful responses**

**200 OK**  
The request has succeeded. The meaning of the success depends on the HTTP method:

- GET: The resource has been fetched and is transmitted in the message body.
- HEAD: The entity headers are in the message body.
- PUT or POST: The resource describing the result of the action is transmitted in the message body.
- TRACE: The message body contains the request message as received by the server.

**201 Created**  
The request has succeeded and a new resource has been created as a result. This is typically the response sent after POST requests, or some PUT requests.

**202 Accepted**  
The request has been received but not yet acted upon. It is noncommittal, since there is no way in HTTP to later send an asynchronous response indicating the outcome of the request. It is intended for cases where another process or server handles the request, or for batch processing.

**203 Non-Authoritative Information**  
This response code means the returned meta-information is not exactly the same as is available from the origin server, but is collected from a local or a third-party copy. This is mostly used for mirrors or backups of another resource. Except for that specific case, the "200 OK" response is preferred to this status.

**204 No Content**  
There is no content to send for this request, but the headers may be useful. The user-agent may update its cached headers for this resource with the new ones.

**205 Reset Content**  
Tells the user-agent to reset the document which sent this request.

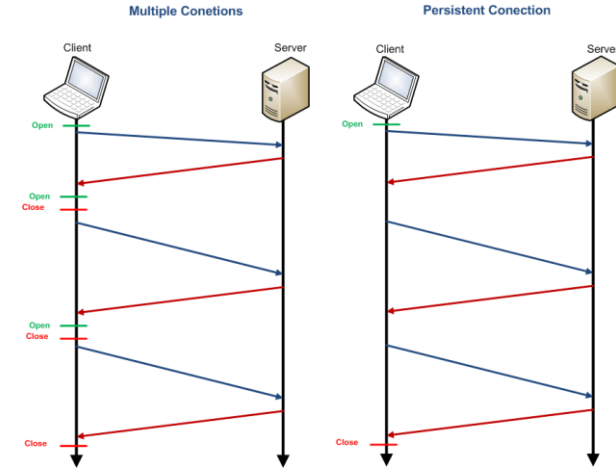
- **Szyfrowanie – rozszerzenie w ramach HTTPS**

## • Persystencja połączeń

- Wprowadzenie mechanizmu KeepAlive
  - Wykorzystywanie jednego połączenia TCP do przesyłania wielu zapytań/odpowiedzi HTTP
  - Brak konieczności zestawiania TCP 3-way-handshake dla każdego requestu
  - Po odesłaniu odpowiedzi serwer nie zrywa połączenia TCP (pozostaje w trybie OPEN)
  - Standardowo 5 sekund (Apache httpd >= 2.2)

## - Dodatkowe mechanizmy przyspieszające transmisje

- Chunked transfer encoding – przesyłanie strumieniowe zamiast buforowanego
- HTTP Pipelining – wysyłanie wielu requestów przed odebraniem pojedynczych odpowiedzi
- Byte serving – transmisja konkretnego fragmentu danych z serwera specyfikowanego przez klienta



Store-and-Forward



Chunked Encoding

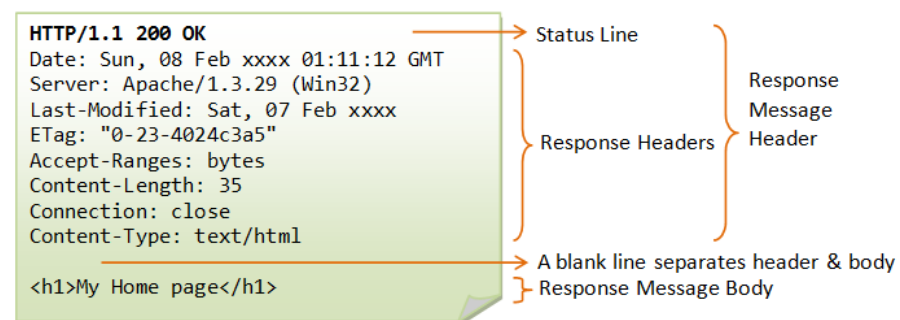
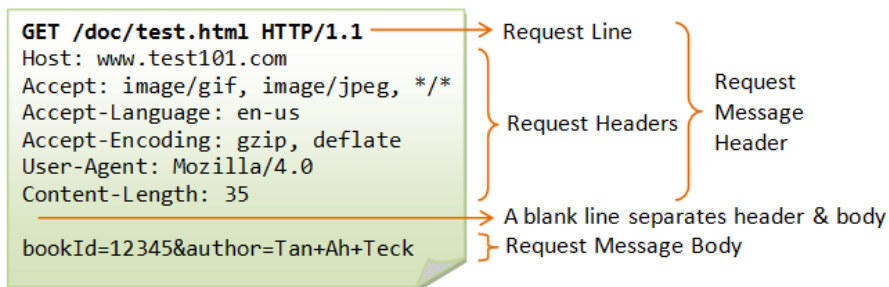






# HTTP – format wiadomości

- **Komunikaty w formie tekstu w kodzie ASCII**
- **Request message (żądanie)**
  - Linia żądania, np.: *GET /images/logo.png HTTP/1.1*
  - Nagłówek żądania (np.: język, kodowanie)
  - Pusta linia
  - Opcjonalna treść żądania
- **Response message (odpowiedź)**
  - Linia statusu (kod, treść), np.: *HTTP/1.1 200 OK*
  - Nagłówki odpowiedzi
  - Pusta linia
  - Opcjonalna treść odpowiedzi





## HTTP – przykładowe żądanie (request)

Linia żądania	Znaczenie
GET / HTTP/1.1	prośba o zwrócenie dokumentu o URI / zgodnie z protokołem HTTP 1.1
Host: example.com	wymagany w HTTP 1.1 nagłówek Host służący do rozpoznania hosta, jeśli serwer na jednym IP obsługuje kilka VirtualHostów
User-Agent: Mozilla/5.0 (X11; U; Linux i686; pl; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7	nazwa aplikacji klienckiej
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8	akceptowane (bądź nieakceptowane dla q=0) przez klienta typy plików
Accept-Language: pl,en-us;q=0.7,en;q=0.3	preferowany język strony – nagłówek przydatny przy Language negotiation
Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7	preferowane kodowanie znaków
Keep-Alive: 300	czas, jaki klient chce zarezerwować do następnego zapytania w przypadku połączenia Keep-Alive
Connection: keep-alive	chęć nawiązania połączenia stałego Keep-Alive z serwerem HTTP/1.0
	znak powrotu karetki i nowej linii (CRLF)



# HTTP – przykładowa odpowiedź (response)

Linia odpowiedzi	Znaczenie
HTTP/1.1 200 OK	kod odpowiedzi HTTP - zaakceptowanie i zwrócenie zawartości
Date: Thu, 20 Dec 2001 12:04:30 GMT	czas serwera
Server: Apache/2.0.50 (Unix) DAV/2	opis aplikacji serwera
Set-Cookie: PSID=d6dd02e9957fb162d2385ca6f2829a73; path=/	nakazanie klientowi zapisania ciasteczka
Cache-Control: no-store, no-cache, must-revalidate	no-store zabrania przechowywania dokumentu na dysku, must-revalidate nakazuje bezwzględnie sprawdzić świeżość dokumentu za każdym razem
Keep-Alive: timeout=15, max=100 Connection: Keep-Alive	akceptacja połączenia Keep-Alive dla klientów HTTP/1.0
Transfer-Encoding: chunked	typ kodowania zawartości stosowanej przez serwer
Content-Type: application/xhtml+xml; charset=utf-8	typ MIME i strona kodowa zwróconego dokumentu
	znak powrotu karetki i nowej linii (CRLF)
<html><head><title>An Example page</title></head> <body> Hello World</body></html>	tutaj zawartość dokumentu (wiele linii)



AGH

# HTTPS – szyfrowana wersja HTTP

- Rozszerzenie HTTP w celu:
  - Autentyfikacji strony WWW
  - Zapewnienia prywatności danych (dostępu)
  - Zapewnieniu integralności danych w procesie transmisji (zapobieganie atakom typu Man-In-The-Middle)
- Wymaga Certyfikatów Cyfrowych po stronie serwera
  - kiedyś kosztowne – metody płatności
  - Od 2016 dostępne także darmowo – Let's encrypt (ISRG)
- Metody szyfrowania
  - Pierwotnie oparte o SSL
  - Aktualnie w oparciu o TLS
- Działa na porcie 443 (protokół TCP), w przeglądarce jako https://
- Sposób działania
  - Wymiana kluczy TLS
  - Żądanie HTTP zakodowane w kanale TLS
- Pierwotnie problem z serwowaniem wielu domen (różne certyfikaty)
- Rozwiązany przez mechanizm SNI – Server Name Identification
- Dostęp do adresu IP domeny czy portu serwera z poziomu nagłówków IP/TCP

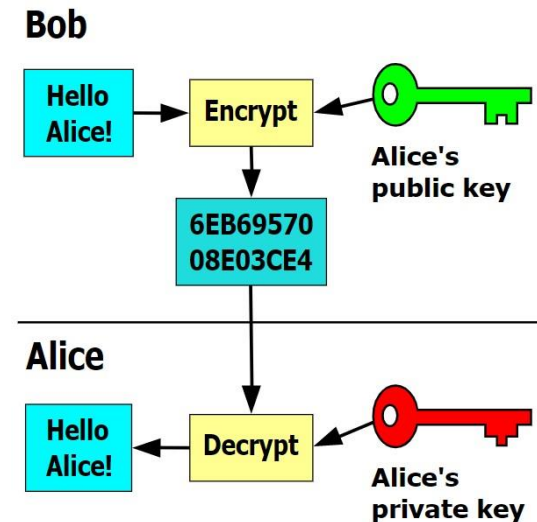




AGH

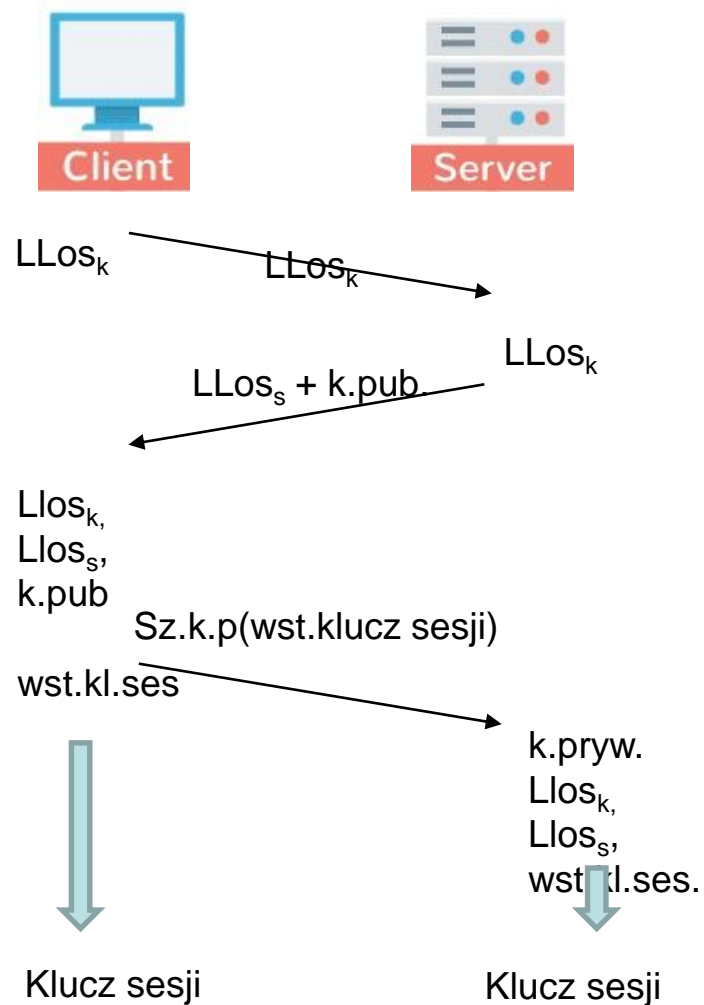
# TLS – protokół szyfrowania

- Transport Layer Security
- Rozwinięcie protokołu SSL (Secure Socket Layer 1994-1999) – SSL 3.1 -> TLS1.0
- NIE JEST ALGORYTMEM SZYFRUJĄCYM
- Zestaw algorytmów, technik i schematów szyfrujących
- Działa w warstwie prezentacji, może zabezpieczać protokoły z warstwy aplikacji (HTTP, POP3, IMAP, FTP, NNTP, SIP)
- Zapewnia
  - Poufność i integralność transmisji
  - Uwierzytelnianie serwera
- Opiera się na
  - Szyfrowaniu asymetrycznym
    - Klucz publiczny – wykorzystywany do zaszyfrowania wiadomości
    - Klucz prywatny – wykorzystywany do odszyfrowania wiadomości
  - Certyfikatach X.509 – Infrastruktura klucza publicznego (Urzędy Certyfikacji)
- Wykorzystuje (TLS3.0) algorytmy/protokoły: AES, DES, 3DES, IDEA, RC2, RC4, RSA, DSS, Diffiego-Hellmana.



# TLS (SSL) – zasada zestawiania bezpiecznego kanału komunikacji

Komunikaty	Opis
K → S ClientHello	Klient wysyła do serwera zgłoszenie zawierające m.in. obsługiwaną wersję protokołu SSL, dozwolone sposoby szyfrowania i kompresji danych, identyfikator sesji oraz <b>liczbę losową</b> używaną potem przy generowaniu klucza
K ← S ServerHello	Serwer odpowiada komunikatem, w którym zwraca m.in.: wersję protokołu SSL, rodzaj szyfrowania i kompresji, oraz <b>liczbę losową</b> .
K ← S Certificate	Serwer wysyła swój certyfikat pozwalając klientowi na sprawdzenie swojej tożsamości (ten etap jest opcjonalny, ale w większości przypadków występuje)
K ← S ServerKeyExchange	Serwer wysyła informację o swoim <b>kluczu publicznym</b> . Rodzaj i długość tego klucza jest określony przez typ algorytmu przesłany w poprzednim komunikacie.
K ← S ServerHelloDone	Serwer zawiadamia, że klient może przejść do następnej fazy zestawiania połączenia.
K → S ClientKeyExchange	Klient wysyła serwerowi <b>wstępny klucz sesji</b> , zaszyfrowany za pomocą klucza <b>publicznego serwera</b> .  Na podstawie ustalonych w poprzednich komunikatach <b>dwóch liczb losowych (klienta i serwera)</b> oraz ustalonego przez klienta <b>wstępnego klucza sesji</b> obie strony generują <b>klucz sesji</b> używany do faktycznej wymiany danych. Uwaga: wygenerowany klucz jest kluczem <b>algorytmu symetrycznego</b> (typowo DES)! Jest on jednak ustalony w sposób bezpieczny i znany jest tylko komunikującym się stronom
K → S ChangeCipherSpec	Klient zawiadamia, że serwer może przełączyć się na komunikację szyfrowaną.
K → S Finished	... oraz że jest gotowy do odbierania danych zaszyfrowanych
K ← S ChangeCipherSpec	Serwer zawiadamia, że wykonał polecenie – od tej pory wysyłał będzie tylko zaszyfrowane informacje...
K ← S Finished	...i od razu wypróbuje mechanizm – ten komunikat jest już wysyłany bezpiecznym kanałem ( <b>symetrycznym</b> )



- **Domain name system**
  - Protokół umożliwiający tłumaczenie nazwy mnemonicznej na odpowiadający jej adres IP
  - Określa zasady komunikacji między:
    - Klientem a serwerem – protokół UDP, port 53
    - Serwerami (np. master -> slave) – protokół TCP, port 53
- **Omawiany na osobnym wykładzie**



AGH

AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE

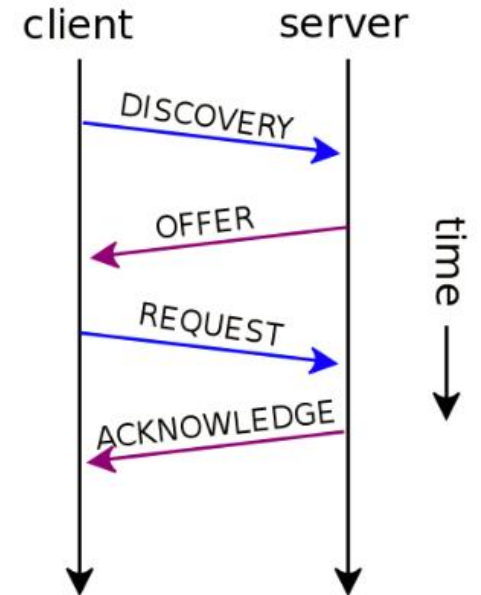
## Sieci komputerowe

Domain Name System

dr inż. Andrzej Opaliński  
andrzej.opalinski@agh.edu.pl

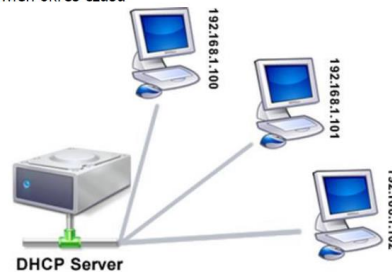
[www.agh.edu.pl](http://www.agh.edu.pl)

- **Dynamic Host Configuration Protocol**
  - Protokół zarządzania siecią w stosie TCP/IP
  - Model bezpołączeniowy, protokół UDP
    - Port 67 (IPv4) dla serwera (546 IPv6)
    - Port 68 (IPv4) dla klienta (547 IPv6)
- **Omawiany na wykładzie dotyczącym warstwy sieciowej**



## Protokół DHCP

- Dynamic Host Configuration Protocol – ulepszona wersja BOOTP
  - RFC 2131 - 1993 r. – publikacja standardu dotyczącego adresu IP
  - RFC 2132 – rozszerzone parametry konfiguracyjne
  - RFC 3315 – wersja dla IPv6
- Tryby przydzielania adresów IP
  - Allokacja ręczna – przydział przez administratora,
  - Allokacja dynamiczna – przydział przez serwer
  - Dzierżawa – przydział dynamiczny na pewien okres czasu
- Używa protokołu UDP
  - IPv4 porty 67 i 68 (jak BOOTP)
  - IPv6 porty 546 i 547





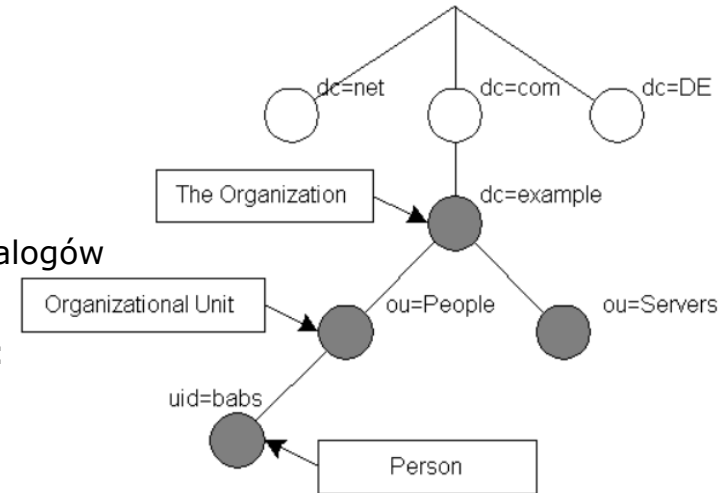


- **Lightweight Directory Access Protocol**

- Protokół umożliwiający korzystanie z usług katalogowych
- Oparty na standardzie X.500
- Pracuje zwykle w oparciu o TCP/IP
- Dane grupowane w strukturze przypominającej drzewo katalogów

- **Usługa katalogowa (Directory Service)**

- Baza danych (umieszczona na serwerze nazw) zawierająca:
  - Użytkowników
  - Aplikacje
  - Urządzenia sieciowe
  - Zasoby sieciowe
- Zwykle baza obiektowa z reprezentacją obiektów, umożliwiającą zarządzanie relacjami pomiędzy nimi
- **Zapewnia logiczny, precyzyjny i hierarchiczny sposób opisu zasobów**
- **Zapewnia bezpieczeństwo kontrolując dostęp do zasobów**
- Przykładowe systemy: Active Directory (Windows), Novell eDirectory, OpenLDAP



- **Atrybuty**

- UID (ang. User Identifier) – identyfikator użytkownika
- RID (ang. Relative Identifier) – liczba reprezentująca względny identyfikator użytkownika
- CN (ang. Common Name) – imię
- SN (ang. Surname) – nazwisko
- OU (ang. Organizational Unit) – jednostka organizacyjna
- O (ang. Organization) – jednostka lub organizacja
- DC (ang. Domain Component) – składnik nazwy domenowej

## • Wpisy i klasy

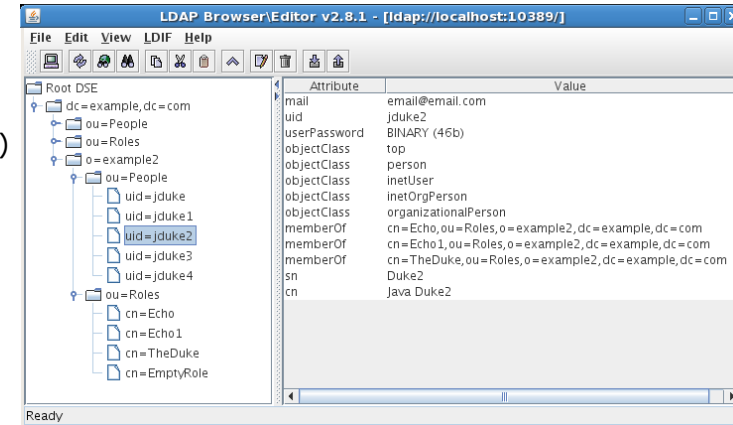
- Informacje przechowywane w formie wpisów (Entries)
  - Każdy wpis jest obiektem jednej z klas (mogą być dziedziczone)
  - Każda z klas składa się z jednego lub wielu atrybutów
- Atrybuty
  - Istnieje wiele typów podstawowych
  - Mogą mieć więcej niż jedną wartość
- Można tworzyć własne klasy i atrybuty
- Wszystkie wpisy są identyfikowalne:
  - Jednoznacznie przez - DN – Distinguished Name
  - W odniesieniu do kontekstu - RDN – Relational Distinguished name
- Dostęp do wpisu (kontekstu, atrybutu,...) jest chroniony przez Acces Control List

## • Operacje LDAP (poprzez klienta)

- **bind – uwierzytelnienie - powiązanie obiektu LDAP użytkownika z połączeniem sieciowym i sesją.**
- unbind – zakończenie sesji i połączenia sieciowego LDAP. Nie potrzebuje potwierdzenia.
- search – żądanie wyszukiwania zasobu,
- modify – modyfikowanie wpisów w bazie danych znajdującej się na serwerze
- add – żądanie dodania wpisów do katalogu, zmiany istniejącego wpisu oraz zmiany nazwy wpisu.
- delete – żądanie usunięcia wpisów z katalogu.

## • Rodzaje uwierzytelniania

- anonimowe – w oparciu o wbudowane konto „guest”
- na hasło – użytkownik loguje się poprzez podanie DN i hasła,
- z użyciem SSL – następuje wymiana certyfikatów,
- z użyciem PROXY – stosowane w Active Directory firmy Microsoft





• **Server Message Block**

- Protokół służący udostępnianiu zasobów komputerowych (plików, drukarek)
- Znany także jako CIFS (Common Internet File System)
- Protokół typu klient-serwer
- Opracowany w latach 80 w IBM
- Jest podstawą „Otoczenia sieciowego” w systemach Windows
- Samba – najpopularniejszy darmowy serwer plików i drukarek oparty na prot. SMB

• **Szczegóły techniczne**

- W warstwie sesji modelu OSI korzysta z protokołu NetBIOS
- Wykorzystuje porty
  - 445 TCP
  - opcjonalne (NetBIOS) 137/UDP, 138/UDP, 139/TCP

OSI					TCP/IP
Application	SMB				Application
Presentation	NetBIOS	NetBEUI	NetBIOS	NetBIOS	
Session	IPX <sup>1</sup>		DECnet	TCP&UDP	TCP/UDP
Transport				IP	IP
Network	802.2, 802.3,802.5	802.2 802.3,802.5	Ethernet V2	Ethernet V2	Ethernet or others
Link					
Physical					

• **Modele bezpieczeństwa**

- Share level – zabezpieczenie dostępu do zasobu poprzez hasło
- User level – prawa dostępu do zasobu dla konkretnych użytkowników

• **Zarządzanie autoryzacją**

- Grupa robocza – każdy z komputerów przechowuje dane lokalnie. Identyfikacja i autoryzacja przebiega lokalnie.
- Domena – podejście scentralizowane oparte o kontroler domeny.



AGH

## SMB - szczegóły

- **Podstawowe operacje**

(w pierwszej wersji – Core Protocol)

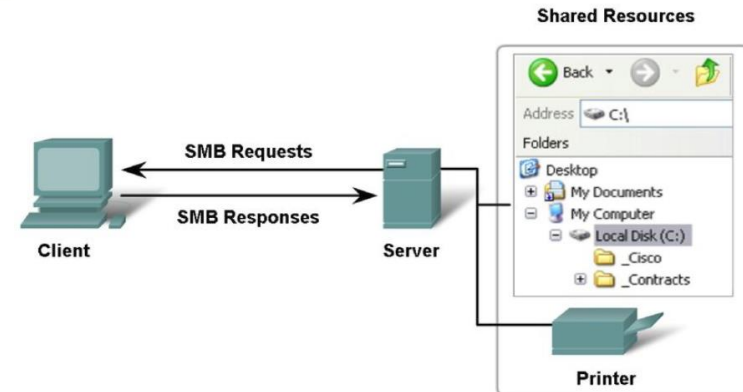
- otwieranie i zamykanie plików,
- otwieranie i zamykanie drukarek,
- odczyt i zapis z/do plików,
- tworzenie i kasowanie plików/katalogów,
- przeszukiwanie katalogów,
- ustawianie i odczytywanie atrybutów plików,
- blokowanie i udostępnianie wybranych fragmentów plików.

- **Kolejne implementacje protokołu**

- PC NETWORK PROGRAM 1.0
- Core Plus Protocol,
- Lan Manager 1.0 (LANMAN 1.0),
- LANMAN 2.0,
- LANMAN 2.1 (Windows for Workgroups),
- CIFS 1.0/ NT Lan Manager 1.0

- **Wersje protokołu**

- SMB/CIFS/SMB1 – od 1990r – DOS, OS/2
- SMB 2.0 – 2006r. – Windows Vista, OS X 10.9
- SMB 2.1 – 2009r. – Windows 7
- SMB 3.0 – 2012r – Windows 8, Windows Server 2012
- SMB 3.1.1 – 2015 – Windows 10, Windows Server 2016





**AGH**

## **Literatura, bibliografia**

Parker T: TCP/IP. Helion, 1997. ISBN 83-86718-55-2.

C. Hunt „TCP/IP network administration”; Second Edition, ISBN 1-56592-322-7

Andrew Stuart Tanenbaum: Sieci komputerowe. Helion, 2004. ISBN 83-7361-557-1.

W.Graniszewski, E.Grochocki, G.Świątek, „Usługi sieciowe” E-Studia Informatyczne, <http://ważniak.mimow.edu.pl>

Brian Komar: Administracja sieci TCP/IP dla każdego. Helion, 2000. ISBN 83-7197-189-3.

K.Maćkowiak, „Protokoły zdalnego logowania Telnet i SSH”

Peter E. Egli - „Overview of TFTP, A very simple file transfer protocol for simple and constrained devices”

RFC 5321, Simple Mail Transfer Protocol, J. Klensin, The Internet Society (October 2008)

Hughes, L. Internet e-mail Protocols, Standards and Implementation. Artech House Publishers. ISBN 0-89006-939-5.

OpenLDAP, „Introduction to OpenLDAP Directory Services”

Bolesław Dawidowicz, JBoss Reference Guide: LDAP

RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1

A.H.Alaidi „Application Layer Functionality and Protocols”