helion.pl

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Apache 2.0 dla zaawansowanych



Autor: Peter Wainwright
Tłumaczenie: Robert Gębarowski
ISBN: 83-7197-874-X
Tytuł oryginału: Professional Apache 2
Format: B5, stron: 984

Olbrzymie możliwości i wszechstronność serwera Apache uczyniły go najbardziej rozpowszechnionym serwerem WWW. Kilka miesięcy temu Apache Software Foundation opublikowała nową wersję Apache 2.0. Najnowsza edycja Apache jest lepiej przystosowana do pracy na różnych platform systemowych niż wersja 1.3, dzięki czemu coraz częściej można spotkać Apache pracującego pod kontrolą Windows czy Mac OS. Stało się to możliwe dzięki wprowadzeniu modułów zwielokrotnionego przetwarzania, tzw. MPM (ang. Multiprocessing Module), dostosowanych do właściwości rozmaitych systemów operacyjnych, jak również wprowadzeniu przenośnych bibliotek fazy wykonywania (ang. Apache Portable Runtime). Porównując Apache 2.0 z wcześniejszymi wersjami zauważymy też istotne zmiany w procesie kompilacji i konsolidacji serwera.

Apache 2.0 to nie tylko zaawansowana architektura serwera, ale również liczne udoskonalenia i nowe funkcje. Książka ta stanowi obszerny i wyczerpujący przewodnik po wszelkich nowościach wprowadzonych w wersji 2.0. Znajdziesz w niej także informacje o zmianach wprowadzonych w porównaniu z poprzednimi wersjami.

Zagadnienia omówione w książce:

- Nowy serwer WWW Apache 2.0 oraz sposoby uaktualniania z Apache 1.3
- Nowe funkcje Apache dostępne wersji w 1.3 i proponowane możliwości migracji serwera WWW do nowej wersji Apache 2.0
- Instalacja serwera Apache w oparciu o dystrybucje binarne oraz kompilowanie serwera z kodu źródłowego dla systemów operacyjnych UNIX i Windows
- Bezpieczne i wydajne tworzenie dynamicznej zawartości stron WWW za pomocą skryptów CGI i FastCGI
- Implementacje wirtualnych hostów w ramach serwera Apache w prostym i złożonym modelu, a także masowe tworzenie hostów wirtualnych
- Przystosowywanie serwerów Apache do sprawowania funkcji serwera pośredniczącego; zagadnienia związane z buforowaniem zawartości WWW, odpornością na błędy i testowaniem wydajności, a także tworzenie klastrów serwerów WWW
- Monitorowanie i zabezpieczanie serwerów Apache
- Rozszerzanie możliwości serwera Apache poprzez włączanie dodatkowych modułów do obsługi programów w językach Perl, Python, PHP, Tcl, Java, Ruby i protokole WebDAV

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

O Autorach 13

Wstęp 17

Rozdział 1. Apache i Internet 21

Apache — anatomia serwera WWW 21
Kod źródłowy Apache 21
Licencja Apache 22
Pomoc techniczna do Apache 22
Jak działa Apache 23
Protokół HTTP 28
Żądania i odpowiedzi HTTP 28
Nagłówki HTTP 32
Praca w sieci oraz TCP/IP 33
Definicje 33
Pakiety i kapsułkowanie 34
Komunikaty ACK, NAK i inne 35
Model sieci TCP/IP 36
Protokoły inne niż IP 38
Adresy IP oraz klasy sieci 39
Specjalne adresy IP 39
Maski sieciowe i wybór tras 40
Odszukiwanie usług — dobrze znane porty 41
Sieciowy super serwer — inetd 43
Przyszłość — protokół IPv6 43
Narzędzia sieciowe 45
Wybór sprzętu dla serwera 48
Obsługiwane platformy 48
Podstawowe wymagania serwera 49
Pamięć operacyjna 50
Interfejs sieciowy 51
Połączenie internetowe 52
Dysk twardy i kontroler 52
Zestawienie właściwości systemu operacyjnego 53

Nadmiarowość i archiwizacja danych ... 54
Specyficzne rozwiązania sprzętowe ... 55
Niech ktoś zrobi to za nas ... 56

Rozdział 2. Serwer Apache od podstaw ... 59

Instalacja Apache ... 60
Dostępność Apache ... 60
Instalacja Apache z dystrybucji binarnej ... 61
Instalacja Apache z użyciem kodu źródłowego ... 63
Instalacja Apache z gotowych pakietów ... 64
Ręczna instalacja serwera Apache ... 68
Uaktualnianie serwera Apache ... 71
Inne zagadnienia ... 72
Podstawowa konfiguracja serwera ... 73
Decyzje ... 73
Nadrzędny plik konfiguracyjny ... 78
Inne dyrektywy podstawowej konfiguracji ... 79
Zaczynanie, kończenie i wznawianie pracy serwera ... 80
Zaczynanie pracy Apache w systemie UNIX ... 80
Zaczynanie pracy Apache w systemie Windows ... 81
Opcje wywołania Apache ... 83
Wznawianie pracy serwera ... 92
Zatrzymanie serwera ... 94
Automatyczne uruchamianie serwera ... 94
Testowanie serwera ... 98
Testowanie przy pomocy przeglądarki ... 98
Testowanie z wiersza poleceń lub program terminalowego ... 99
Testowanie konfiguracji serwera bez jego uruchamiania ... 101
Informacja o stanie serwera w wierszu poleceń ... 102
Graficzne narzędzia konfiguracyjne ... 103
Comanche ... 103
TkApache ... 106
LinuxConf ... 107
Webmin ... 107
ApacheConf ... 110
Inne narzędzia konfiguracyjne ... 111

Rozdział 3. Budowa Apache według własnych wymagań ... 113

Dlaczego budować Apache samodzielnie ... 113
Budowa Apache z kodu źródłowego ... 115
Konfiguracja i budowa Apache ... 117
Wybór modułów wstawianych do serwera ... 121
Budowa Apache jako serwera dynamicznego ... 125
Zmiana porządku modułów w Apache 1.3 ... 127
Konfiguracja zaawansowana ... 129
Konfiguracja układu Apache ... 129
Wybór modułu MPM ... 136
Reguły ... 138
Tworzenie Apache z obsługą suExec ... 140
Konfiguracja plików pomocniczych i skryptów Apache ... 142
Konfiguracja budowy Apache 2.0 dla wielu platform ... 143
Konfiguracja Apache do produkcji lub do usuwania błędów ... 145

Konfiguracja Apache do dystrybucji binarnej ... 145
Konfiguracja ścieżki bibliotek i plików wstawianych Apache ... 145
Konfiguracja środowiska budowy ... 146
Budowanie modułów przez configure i apxs ... 148
Dodawanie niezależnych modułów za pomocą configure ... 148
Budowanie modułów za pomocą apxs ... 150
Instalacja modułów za pomocą apxs ... 151
Szablony modułów generowane przez apxs ... 152
Przeciążanie i użycie apxs w skryptach makefile ... 153

Rozdział 4. Konfiguracja Apache według własnych wymagań ... 155

Gdzie Apache szuka swojej konfiguracji ... 156
Składnia pliku konfiguracyjnego ... 156
Konfiguracja hostów wirtualnych ... 156
Wstawianie plików konfiguracyjnych ... 157
Konfiguracje katalogów ... 159
Konfiguracja warunkowa ... 159
Struktura konfiguracji serwera Apache ... 162
Dyrektywy kontenerowe serwera Apache ... 164
Typy dyrektyw i lokalizacji ... 167
Gdzie można umieścić dyrektywy ... 170
Zasięg kontenera i zagnieżdżanie ... 172
Jak Apache łączy kontenery i ich zawartość ... 173
Poprawność dyrektyw zawartych w kontenerach ... 174
Opcje i przeciążanie ... 175
Włączanie i wyłączanie przez dyrektywy Options ... 175
Przeciążanie dyrektyw przez konfiguracje katalogów ... 178
Ograniczanie dostępu dyrektywami allow i deny ... 181
Kontrola dostępu w oparciu o nazwę ... 182
Kontrola dostępu w oparciu o adresy IP ... 183
Kontrola dostępu do podsieci przez adres i maskę sieci ... 184
Kontrola dostępu w oparciu o nagłówek HTTP ... 185
Dostęp dla hosta przy uwierzytelnianiu użytkownika ... 186
Przeciążanie dostępu dla hosta ... 187
Wykazy zawartości katalogów ... 187
Włączanie i wyłączanie indeksowania katalogu ... 187
Jak moduł mod_autoindex generuje stronę HTML ... 189
Ukrywanie plików przy pomocy dyrektywy IndexIgnore ... 195
Sterowanie porządkiem sortowania ... 196
Przypisywanie ikon w procesie indeksowania ... 197
Przypisywanie opisów ... 200
Środowisko serwera Apache ... 201
Ustawianie, usuwanie i przekazywanie zmiennych z shella ... 202
Warunkowe ustawianie zmiennych ... 203
Zmienne specjalne dla przeglądarek ... 204
Wykrywanie robotów za pomocą dyrektywy BrowserMatch ... 206
Przekazywanie zmiennych do skryptów CGI ... 206
Kontrola dostępu warunkowego ... 207
SetEnvIf a SetEnv ... 207
Ustawianie zmiennych przez mod_rewrite ... 208
Kontrola nagłówków żądania i odpowiedzi ... 208
Ustawianie własnych nagłówków odpowiedzi ... 210
Ustawianie własnych nagłówków żądania ... 212

Wstawianie wartości dynamicznych do nagłówków ... 212
Ustawianie warunkowe własnych nagłówków ... 213
Wyciąganie nagłówków odpowiedzi z plików metadanych ... 214
Ustawianie ograniczeń czasowych ... 215
Wysyłanie zawartości w oryginalnej postaci ... 218
Kontrola nagłówka identyfikacyjnego serwera ... 219
Wysyłanie skrótu treści ... 221
Pomoc sąsiedzka ... 221
Sterowanie robotami za pomocą pliku robots.txt ... 222
Sterowanie robotami w języku HTML ... 223
Sterowanie robotami za pomocą kontroli dostępu ... 224
Przyciąganie uwagi robotów ... 224
Jak zapewnić robotom właściwą informację ... 225
Znane roboty, złe roboty i dalsza lektura ... 226

Rozdział 5. Czego potrzebuje klient ... 227

Uzgadnianie i obsługa zawartości ... 227
Typy plików ... 228
Kodowanie pliku ... 232
Wersje językowe plików ... 237
Zestaw znaków dla pliku ... 238
Uzgadnianie zawartości ... 241
Uzgadnianie zawartości z pomocą MultiViews ... 243
Permutacje plików i adresy URL zgodne z MultiViews ... 250
Magiczne typy MIME ... 254
Obsługa błędów i odpowiedzi ... 259
Jak serwer Apache obsługuje błędy ... 259
Błędy i kody odpowiedzi ... 259
Dyrektywa ErrorDocument ... 260
Ograniczenia dyrektywy ErrorDocument ... 264
Nazwy zastępcze i przeadresowanie ... 265
Nazwy zastępcze dla lokalizacji plików i skryptów ... 266
Przeadresowanie ... 268
Podstawianie adresów URL z pomocą modułu mod_rewrite ... 271
Mapy graficzne obsługiwane po stronie serwera ... 293
Dopasowanie adresów URL zawierających błędy pisowni ... 298

Rozdział 6. Tworzenie zawartości dynamicznej ... 301

Wstawki po stronie serwera ... 303
Włączanie SSI ... 303
Format poleceń SSI ... 306
Zestaw poleceń SSI ... 307
Zmienne SSI ... 307
Przekazanie końca ścieżki do dokumentów składanych przez serwer ... 309
Ustawienie formatu dla daty i błędu ... 309
Stosowanie szablonów z pomocą wstawek SSI ... 310
Buforowanie dokumentów składanych przez serwer ... 312
Identyfikacja na podstawie praw wykonania dokumentów składanych przez serwer ... 313
Protokół CGI ... 314
Skrypt CGI i środowisko ... 315
Konfiguracja serwera Apache pod kątem rozpoznawania skryptów CGI ... 317
Wyzwalanie skryptów CGI poprzez zdarzenia ... 322

Tworzenie skryptów CGI i usuwanie błędów ... 325
Skrypt CGI w wersji mini ... 326
Skrypty interakcyjne — prosty formularz ... 330
Dodawanie nagłówków ... 331
Usuwanie błędów ze skryptów CGI ... 332
Ustawienie gniazda dla demona CGI ... 338
Ograniczenie wykorzystania zasobów CGI ... 339
Procedury działania i obsługi oraz filtry ... 340
Procedury obsługi ... 341
Filtry ... 348
Dynamiczna zawartość a bezpieczeństwo ... 353
Zagadnienia bezpieczeństwa skryptów CGI ... 353
Doradztwo dotyczące spraw bezpieczeństwa WWW ... 354
Zagadnienia bezpieczeństwa związane z konfiguracją Apache dla skryptów CGI ... 355
Przykład stwarzającego zagrożenia skryptu CGI ... 356
Niebezpieczne skrypty CGI ... 360
Otoczki CGI ... 361
Wykaz niezbędnych środków zapobiegawczych ... 372
Ulepszone skrypty CGI — FastCGI ... 373

Rozdział 7. Sprawowanie funkcji hosta dla wielu witryn WWW ... 391

Implementacja katalogów użytkownika przez UserDir ... 392
Włączanie i wyłącznie określonych użytkowników ... 394
Przeadresowanie użytkowników na inne serwery ... 395
Inne metody implementacji katalogów użytkownika ... 395
Odrębne serwery ... 396
Zawężenie perspektywy serwera Apache ... 397
Określanie różnych konfiguracji oraz katalogów głównych serwera ... 398
Uruchamianie odrębnych serwerów z tym samym plikiem konfiguracyjnym ... 399
Dzielenie zewnętrznych plików konfiguracyjnych ... 400
Definiowane przez adres IP hosty wirtualne ... 401
Wielokrotne adresy IP, odrębne sieci, wirtualne interfejsy ... 401
Konfiguracja nasłuchu serwera Apache ... 403
Definicje hostów wirtualnych na bazie adresu IP ... 404
Hosty wirtualne a konfiguracja na poziomie serwera ... 407
Określenie praw użytkownika hosta wirtualnego ... 408
Wykluczone dyrektywy ... 412
Domyślne hosty wirtualne ... 413
Definiowane przez nazwę hosty wirtualne ... 414
Definiowanie nazwanych hostów wirtualnych ... 415
Nazwy serwerów i ich nazwy zastępcze ... 416
Definiowanie domyślnego hosta dla potrzeb nazwanego hosta wirtualnego ... 417
Mieszanie funkcji hostów wirtualnych opartych na adresie IP i nazwie ... 417
Problemy z hostami wirtualnymi ... 420
Pliki dzienników i uchwyty plików ... 420
Hosty wirtualne a bezpieczeństwo serwera ... 423
Bezpieczny protokół HTTP a hosty wirtualne ... 424
Obsługa klientów HTTP/1.0 za pomocą hostów wirtualnych definiowanych na bazie nazwy ... 425
Dynamiczne sprawowanie funkcji hosta ... 427
Masowe sprawowanie funkcji hosta za pomocą nazw zastępczych hostów wirtualnych ... 427
Dynamiczne odwzorowanie nazw hostów za pomocą modułu mod_rewrite ... 434
Tworzenie i włączanie plików konfiguracyjnych w locie za pomocą modułu mod_perl ... 435

Rozdział 8. Poprawa wydajności Apache 441
Dyrektywy wpływające na wydajność Apache 442
Konfiguracja modułów MPM — procesy i wątki 443
Dyrektywy poprawiające wydajność protokołów sieciowych 454
Dyrektywy związane z wydajnością protokołu HTTP 456
Dyrektywy nakładające ograniczenia na żądania HTTP 459
Konfiguracja Apache dla większej wydajności 461
Dyrektywy istotne dla wydajności 461
Dodatkowe dyrektywy dostrajania wydajności 466
Testowanie wydajności serwera Apache 473
Testy wydajności Apache za pomocą programu ab 473
Narzędzia oceny wydajności napisane niezależnie 478
Strategia badania wydajności i pułapki z tym związane 478
Lista czynności przy zwiększaniu wydajności 479
Apache w roli serwera pośredniczącego 480
Instalacja i aktywacja usług serwera pośredniczącego 481
Zwykłe operacje pośredniczenia 482
Konfiguracja Apache jako serwera pośredniczącego 482
Dopasowanie URL przez dyrektywy kontenerowe 484
Blokada witryn przez serwer pośredniczący 486
Lokalizacja zdalnych zasobów URL i ukrycie serwerów 487
Przekazywanie żądań do zdalnych serwerów pośredniczących 490
Łańcuchy serwerów pośredniczących i nagłówek Via 491
Serwery pośredniczące a sieć intranet 494
Obsługa błędów 495
Przedawnienie żądań do serwerów pośredniczących 496
Tunelowanie innych protokołów 497
Dostrajanie operacji w serwerach pośredniczących 498
Serwer pośredniczący Squid 498
Buforowanie 499
Aktywacja buforowania 499
Buforowanie w plikach 500
Buforowanie w pamięci (tylko serwer Apache 2.0) 503
Koordynacja pamięci bufora i bufora dyskowego 504
Ogólna konfiguracja bufora 504
Współpraca z buforami zewnętrznymi 509
Tworzenie klastrów i odporność na uszkodzenia 511
Serwer zapasowy z użyciem przeadresowana drugorzędnego systemu DNS 512
Rozłożenie obciążenia za pomocą karuzeli DNS 513
Serwer zapasowy z użyciem pływającego adresu IP 514
Sprzętowe równoważenie obciążenia 514
Organizacja klastrów z serwerem Apache 515
Inne rozwiązania organizacji klastrów 518

Rozdział 9. Monitorowanie serwera Apache 521
Dzienniki i rejestracja zdarzeń 521
Pliki dzienników a bezpieczeństwo 522
Dziennik błędów 523
Rejestracja błędów w dzienniku systemowym 524
Dzienniki przekazu 527
Obróbka plików dziennika przy pomocy rozmaitych aplikacji 536
Rotacja dzienników 538

Fałszerstwa, dzienniki i statystyka ... 541
Czego nie można dowiedzieć się na podstawie analizy dzienników serwera WWW ... 542
Analog — narzędzie do analizy dzienników ... 543
Informacja o serwerze ... 560
Informacja o stanie serwera ... 560
Informacja o serwerze ... 563
Zabezpieczenie dostępu do informacji o serwerze ... 565
Obserwacja aktywności użytkownika ... 566
Alternatywy dla obserwacji użytkownika ... 567
Obserwacje prowadzone z wykorzystaniem cookie i modułu mod_usertrack ... 567
Obserwacje adresu URL za pomocą modułu mod_session ... 572
Inne opcje monitoringu sesji ... 577

Rozdział 10. Zabezpieczenie serwera Apache ... 579

Uwierzytelnienie użytkownika ... 579
Moduły uwierzytelniające serwera Apache ... 580
Wymagania dla konfiguracji uwierzytelnienia ... 582
Użycie dyrektyw uwierzytelniających w plikach .htaccess ... 584
Uwierzytelnienie podstawowe ... 584
Uwierzytelnienie przez skrót ... 586
Uwierzytelnienie anonimowe ... 588
Ustawienie informacji o użytkownikach ... 589
Określenie wymagań użytkownika ... 597
Łączenie schematów uwierzytelnienia ... 599
Łączenie uwierzytelnienia użytkownika i hosta ... 601
Zabezpieczenie przez SSL uwierzytelnienia podstawowego ... 602
SSL a serwer Apache ... 603
Pobranie OpenSSL i ModSSL ... 604
Kompilacja oraz instalacja biblioteki OpenSSL ... 604
Tworzenie oraz instalacja modułu mod_ssl dla Apache 2.0 ... 608
Tworzenie oraz instalacja modułu mod_ssl dla Apache 1.3 ... 608
Podstawowa konfiguracja protokołu SSL ... 612
Instalacja klucza prywatnego ... 614
Żądanie poświadczonego oraz tymczasowego certyfikatu ... 615
Uzyskiwanie poświadczonego certyfikatu ... 617
Zaawansowana konfiguracja protokołu SSL ... 619
Konfiguracja SSL na poziomie serwera ... 619
Certyfikaty klienta ... 632
Zastosowanie certyfikacji klienta w połączeniu z uwierzytelnianiem użytkownika ... 634
Dziennik transakcji SSL ... 635
Zmienne środowiskowe protokołu SSL a skrypty CGI ... 637
Protokół SSL i hosty wirtualne ... 640
Właściwości zaawansowane oraz eksperymentalne ... 642

Rozdział 11. Wzmocnienie zabezpieczeń serwera WWW ... 645

Właściwości serwera Apache ... 645
Niechciane pliki ... 646
Automatyczne indeksowanie katalogów ... 647
Dowiązania symboliczne ... 648
Pliki włączane po stronie serwera ... 649
Zwroty serwera ... 649
Katalogi użytkownika ... 650

Prawa dostępu do plików ... 650
Przeglądanie informacji o serwerze za pomocą modułu mod_info ... 651
Ograniczenie przywilejów serwera ... 652
Ograniczenie dostępu na podstawie nazwy hosta i adresu IP ... 652
Inne środki bezpieczeństwa ... 654
Serwer wyspecjalizowany ... 655
Nienaruszalność plików ... 656
Narzędzie md5sum ... 657
Pakiet narzędziowy Tripwire ... 658
Hartowanie serwera ... 659
Minimalizacja usług ... 659
Skanowanie portów narzędziem Nmap ... 661
Próbkowanie portów programem Nessus ... 662
Hartowanie systemu Windows 2000 ... 662
Wyłączanie usług sieciowych ... 663
Protokół FTP ... 663
Usługa telnet ... 663
Usługi rlogin, rsh, rexec i rcp ... 664
Sieciowy system plików NFS ... 664
Usługa sendmail i inne środki transportu poczty elektronicznej (MTA) ... 664
Ograniczanie dostępu do usług z pomocą otoczek TCP ... 665
Usuwanie luk w zabezpieczeniach, alerty i zasoby online ... 666
Lista FAQ na temat bezpieczeństwa serwerów WWW ... 667
Lista korespondencyjna i archiwum BugTraQ ... 667
Raporty dla systemów operacyjnych ... 667
Powiadamianie o uaktualnieniach pakietów i modułów ... 667
Usunięcie cennych danych z serwera ... 668
Uaktywnienie bezpiecznych rejestracji z pomocą SSH ... 668
Kompilacja, konsolidacja i instalacja pakietu OpenSSH ... 670
Strategie uwierzytelniania ... 672
Konfiguracja protokołu SSH ... 673
Testowanie instalacji pakietu SSH ... 677
Rozszerzenie instalacji SSH o uwierzytelnianie użytkowników ... 678
Bezpieczne kopie zapasowe serwera tworzone z wykorzystaniem uwierzytelnienia typu Rsync oraz protokołu SSH ... 678
Przekazywanie połączeń klienta do aplikacji serwera ... 680
Zapory sieciowe i serwery o wielu interfejsach ... 681
Typy zapór sieciowych ... 681
Projektowanie topologii sieci ... 681
Zestawienie wytycznych dla zapewnienia bezpieczeństwa serwera ... 685
Unikaj uruchamiania usług z przywilejami użytkownika root ... 685
Utrzymuj dzienniki serwera w dobrym stanie ... 685
Nie komplikuj ... 686
Blokuj dostęp niepożądanych klientów ... 686
Stwórz sprawny system tworzenia kopii zapasowych ... 687
Zaplanuj wysoką dostępność, pojemność i przywracanie działania po awarii ... 687
Monitoruj serwer ... 688
Zachowaj ostrożność przy przepływie informacji ... 688
Wybierz skuteczną politykę wobec robotów ... 688

Rozdział 12. Rozszerzenia serwera Apache 689
WebDAV 689
Uzupełnienie serwera Apache o protokół WebDAV 690
Protokół WebDAV 691
Konfiguracja serwera Apache dla potrzeb protokołu WebDAV 694
Ograniczenia konfiguracji niezbędne dla bezpiecznej implementacji serwera WebDAV 697
Protokół WebDAV a hosty wirtualne 698
Konfiguracja czasu trwania blokady DAV 698
Ograniczenia repozytoriów opartych na systemie plików 699
Ochrona serwerów WebDAV 700
Zaawansowana konfiguracja protokołu WebDAV 700
Współpraca protokołu WebDAV ze skryptami CGI oraz innymi procedurami obsługi zawartości 703
Perl 704
Tworzenie i instalacja modułu mod_perl 706
Przeniesienie instalacji mod_perl z Apache 1.3 do Apache 2.0 713
Konfiguracja i implementacja procedur obsługi w języku Perl 715
Konfiguracja i implementacja filtrów Perla 728
Ostrzeżenia, tryb skażenia i usuwanie błędów 729
Zarządzanie wątkami Perla w module mod_perl 2 731
Inicjalizacja modułów przy uruchamianiu 736
Wznawianie działania modułu mod_perl oraz moduły ładujące się automatycznie 737
Tworzenie strony z informacją o statusie modułu mod_perl 738
Uruchamianie skryptów CGI w środowisku mod_perl 739
Pułapki skryptów CGI 741
Przekazywanie zmiennych do procedur obsługi napisanych w języku Perl 744
Zastosowanie modułu mod_perl wraz ze wstawkami po stronie serwera 744
Wbudowywanie kodu Perla do dokumentów HTML 745
Wbudowywanie Perla do konfiguracji serwera Apache 751
Python 752
Instalacja Pythona 753
Wymagania wobec instalacji serwera Apache 753
Apache 2.0 i mod_python 754
Konfiguracja modułu mod_python 755
Kompilacja i instalacja modułu mod_python 755
Konfiguracja serwera Apache z modułem mod_python 755
Testowanie modułu mod_python 756
Dyrektywy konfiguracyjne serwera Apache a moduł mod_python 757
Apache 2.0 i mod_snake 758
Pobranie i instalacja narzędzia Swig 759
Instalacja modułu mod_snake dla serwera Apache 2.0 759
Kompilacja i instalacja modułu mod_snake 760
Konfiguracja serwera Apache 2.0 z modułem mod_snake 760
mod_snake_epy 761
mod_snake_cgi 762
mod_snake_simple 762
PHP 763
Instalacja PHP 764
Testowanie instalacji PHP 769
Pliki konfiguracyjne i dyrektywy 770
Plik konfiguracyjny PHP 770
Dyrektywy serwera Apache 771

Ustawienia ... 771
Usuwanie problemów ... 773
Tcl ... 774
Tworzenie i instalacja serwera Apache z modułem mod_tcl ... 775
Konfiguracja modułu mod_tcl w plikach konfiguracyjnych serwera Apache ... 777
Tworzenie skryptów dla modułu mod_tcl ... 779
Interfejs programowania aplikacji serwera Apache w module mod_tcl ... 780
Java ... 787
Podstawowa instalacja i konfiguracja ... 788
Pliki konfiguracyjne Tomcata ... 794
Dyrektywy konfiguracyjne z pliku server.xml ... 794
Konfiguracja Tomcata jako samodzielnego serwera WWW ... 801
Konfiguracja serwera Tomcat do pracy z serwerem Apache ... 802
Złącze APJ ... 803
Złącze WARP ... 807
Konfiguracja bezpiecznych połączeń SSL ... 808
Konfiguracja obsługi serwera pośredniczącego ... 810
Ruby ... 810
Kompilacja i instalacja ... 811
Konfiguracja serwera Apache dla modułu mod_ruby ... 812
Wykorzystanie eRuby ... 814
cgi.rb ... 815
Buforowanie danych wyjściowych i obsługa wyjątków ... 816
Tablice serwera Apache ... 818
Obiekt żądania serwera Apache ... 819
Dyrektywy konfiguracyjne modułu mod_ruby ... 832
Tworzenie procedur obsługi dla modułu mod_ruby ... 833
Moduł mod_ruby i bezpieczeństwo ... 836

Dodatek A Przydatne dokumenty RFC ... 839

Dodatek B Warianty serwera Apache ... 843

Dodatek C Licencja oprogramowania Apache ... 845

Dodatek D Zmienne środowiskowe ... 847

Dodatek E Wstawki SSI ... 853

Dodatek F Wyrażenia regularne ... 861

Dodatek G Niezależnie napisane moduły Apache ... 865

Dodatek H Nagłówki HTTP i kody stanu ... 871

Dodatek I Wykaz dyrektyw uszeregowanych według modułów ... 887

Dodatek J Alfabetyczny spis dyrektyw ... 915

Skorowidz ... 937

1

Apache i Internet

Niniejszy rozdział stanowi wprowadzenie do tego, co stanowi zasadniczy temat książki — do serwera Apache i protokołu HTTP. Jest to fragment książki przeznaczony dla nowicjuszy w pracy z serwerem Apache i osób nie mających doświadczenia z serwerami WWW. Większość przestawionych tu pojęć jest dobrze znana tym Czytelnikom, którzy zdobyli już pewne doświadczenie w administrowaniu systemem i są oczytani w problemach internetowych; z tego powodu mogą śmiało pominąć rozdział 1. i przejść bezpośrednio do rozdziału 2.

Aby serwer Apache mógł działać, wymaga oczywiście komputera. W tym rozdziale omówimy wybór sprzętu dla komputera, na którym ma być uruchomiony serwer Apache oraz najważniejsze kryteria tego wyboru. Chociaż ręczna instalacja serwera Apache nie sprawia żadnych trudności, to z myślą o Czytelnikach szukających gotowych rozwiązań wspartych obsługą dostawcy zajmiemy się wariantami systemów komputerowych, przeznaczonych specjalnie dla serwera Apache. Na zakończenie przyjrzymy się niektórym dostępnym w instalacjach serwera Apache narzędziom do konfiguracji graficznej.

Apache — anatomia serwera WWW

W tym podrozdziale zaprezentowanych zostanie kilka podstawowych pojęć ułatwiających zrozumienie, czym właściwie jest *serwer WWW* i jaka jest jego rola. Omówione zostaną — w podstawowym zakresie — praca serwera i efekty jego działania. Zastanowimy się także nad przyczynami ciągle rosnącej popularności serwera Apache, który utrwala swoją pozycję wśród serwerów WWW w Internecie.

Kod źródłowy Apache

Serwer Apache zdobył w Internecie pozycję najpopularniejszego serwera WWW. Sekret tego ogromnego sukcesu tkwi w tym, że jest to projekt o swobodnym dostępie do *kodu źródłowego*, co oznacza, że można bez przeszkód i nieodpłatnie korzystać z kodu źródłowego. Sprzyja to powiększaniu się i różnicowaniu grona osób pracujących z tym serwerem i daje

nieporównywalnie większe możliwości użytkownikom, którzy chcą wzbogacić swój serwer o nowe właściwości. Istotnie, kilka najważniejszych modułów Apache wzięło swój początek z opracowanych niezależnie projektów. Dobrym przykładem na to są chociażby moduły `mod_vhost_alias` i `mod_dav`.

Warto zaznaczyć, że nie tylko umożliwiono korzystanie z kodu źródłowego Apache, ale nawet aktywnie się do użycia tego kodu zachęca! Obecnie wszystkim dystrybucjom binarnym towarzyszą kompletne kopie gotowego do kompilacji kodu źródłowego. Przeczytanie go może być nie tylko użyteczne i pouczające, ale często pozwala odkryć błędy ukryte w programie, co jest wielką siłą otwartej krytyki ze strony ludzi z branży. Po odkryciu błędu każdy może wysłać poprawkę do Internetu lub zespołu programistów rozwijających Apache. Dzięki temu rozwój serwera i tworzonych niezależnie modułów jest szybszy i szybciej naprawia się ujawnione w programie błędy.

Licencja Apache

Kod źródłowy Apache, podobnie jak większość dostępnego w Internecie kodu źródłowego, jest objęty licencją umożliwiającą dystrybucję. Jednak Apache nie posiada licencji GPL — powszechnej licencji GNU., lecz własną licencję. *Licencja dla Apache* jest zdecydowanie mniej restrykcyjna niż GPL i pozwala na swobodniejsze wykorzystanie oprogramowania Apache w aplikacjach komercyjnych, określając jedynie kilka warunków o podstawowym charakterze.

Jeśli zamierzamy korzystać z serwera Apache tylko dla swoich potrzeb, właściwie nie musimy się przejmować treścią licencji. W przypadku Apache będzie nas ona dotyczyć tylko wtedy, gdy planujemy dystrybucję, zmianę nazwy programu, sprzedaż wersji oprogramowania Apache lub produktu, w którego skład wchodzi oprogramowanie serwera. W przypadku wątpliwości należy jednak koniecznie zapoznać się uważnie z treścią licencji. Licencja Apache jest bardzo zwięzła — nie przekracza objętości jednej strony. Jest dołączana do każdej dystrybucji źródłowej i binarnej; dla wygody Czytelnika została także zamieszczona w dodatku C tej książki.

Należy zwrócić uwagę na fakt, że kilka niezależnych produktów, utworzonych na bazie serwera Apache, ma swoje własne licencje; a zatem licencja Apache ich nie dotyczy.

Pomoc techniczna do Apache

Zespół **Apache Software Foundation** nie zapewnia bezpośredniej pomocy technicznej w rozwiązywaniu problemów związanych z użytkowaniem serwera Apache. Jednak, gdy zawiodą wszelkie inne metody, można przekazać tam raport o wykrytych błędach i innych napotkanych trudnościach. Podobnie jak w przypadku innych projektów o otwartym dostępie do kodu źródłowego, najlepszą pomoc ma do zaoferowania społeczność Internetu, stanowiąca nieformalną, ale dobrze poinformowaną grupę. W wielu przypadkach taka pomoc jest wystarczająca. Apache cieszy się dobrą opinią i nieczęsto zdarzają się przypadki wymagające natychmiastowej interwencji.

Serwery Apache nie wymagają tak „awaryjnych napraw", jakich wymagają często niektóre inne serwery WWW zwłaszcza dla Windows. Już sam fakt, że Apache jest bardziej popularny

niż wszystkie inne serwery WWW razem wzięte, pozwala sądzić o jego niezwykłej klasie (statystyczne podsumowanie niektórych testów popularności jest dostępne pod adresem *http://www.netcraft.com/survey/*).

Jeśli jednak uznamy, że wsparcie jest niezbędne, będziemy mieć do wyboru kilka możliwości:

- Linia produktów WebSphere firmy IBM wykorzystuje Apache jako zasadniczą część składową w wersjach na AIX, Linux, Solaris i Windows NT. IBM oferuje pomoc techniczną dla swej wersji Apache o nazwie *IBM HTTPD Server*.
- Firma Apple Computers dołączyła serwer Apache jako standardowy komponent do obydwu swoich systemów operacyjnych dla małych komputerów — MacOS X Server oraz MacOS X. Konstrukcja systemu MacOS X jest oparta na odmianie BSD UNIX; w związku z tym serwer Apache nie różni się właściwie od typowych instalacji na BSD czy Linux.
- Istnieje także oparty na oprogramowaniu Apache serwer v.2.0.0, opracowany przez Hewlett Packard na hp-ux 11.0 i hp-ux 11i (PA-RISC) oczekujący na premierę.

Ponadto inni dostawcy serwerów Apache na Linux (jak na przykład RedHat) oferują wsparcie techniczne — a w ramach tego również obsługę serwerów Apache. Jakość usług, jak to zwykle bywa różna. Na szczęście tam, gdzie chodzi o Linux, nie ma kłopotów z ustaleniem rzetelności dostawcy, ponieważ nigdy nie ma kłopotów ze znalezieniem w sieci osób opiniujących te usługi.

Trzeba jeszcze przypomnieć, że obsługę serwera Apache oferują także niektórzy dostawcy usług internetowych oraz integratorzy systemów. Wykaz takich organizacji znajduje się w witrynie Apache, pod adresem *http://www.apache.org/info/support.cgi*. Liczba dostawców usług internetowych, którzy oferują serwery oparte na oprogramowaniu Apache wzrosła wyraźnie w ostatnich latach, co dotyczy całego wachlarza serwerów — od serwerów dedykowanych i kolokacji, przez serwery wirtualne, do zwykłych gościnnych kont.

Jak działa Apache

Serwer Apache nie działa jak zwykła aplikacja użytkownika, taka jak na przykład edytor tekstu. Działa za kulisami, oferując usługi innym aplikacjom, które chcą się z serwerem komunikować. Przykładem programu, który może komunikować się z serwerem, jest przeglądarka WWW.

> W terminologii stosowanej w systemie UNIX aplikacje, które zapewniają usługi dla użytkownika nie komunikując się z nim bezpośrednio, nazywa się demonami. Serwer Apache działa również jako demon w systemie Windows NT, w którym takie programy noszą miano usług. Systemy Windows 95/98 oraz ME nie są zdolne do uruchomienia serwera Apache jako usługi. W tych systemach musi on być uruchomiony z wiersza poleceń (w oknie MS-DOS albo po wybraniu *Uruchom...* z menu *Start*), choć po uruchomieniu nie dochodzi do żadnych dalszych interakcji serwera z użytkownikiem.

Serwer Apache jest zaprojektowany tak, by pracował poprzez sieć i dlatego aplikacje, które z nim rozmawiają mogą znajdować się na zupełnie innym komputerze. Aplikacje komunikujące się z serwerem poprzez sieć noszą ogólne miano **klientów**. Pojęcie sieci jest oczywiście

szerokie i może dotyczyć zarówno lokalnej sieci intranet, jak i całego Internetu — wszystko zależy od charakteru serwera. W dalszej części tego rozdziału pomówimy o sieciach, jak działają i jak Apache rozmawia za ich pośrednictwem z klientami.

Klientem stosowanym w sieci najczęściej jest przeglądarka internetowa (stąd zwykle używając pojęcia „klient" mamy na myśli właśnie ją). Niemniej jednak istnieje kilka innych ważnych klientów. Najważniejszymi są roboty WWW i pająki indeksujące witryny WWW.

Głównym zadaniem serwera WWW jest przetłumaczenie otrzymanego żądania na właściwą odpowiedź, stosownie do potrzeby chwili. Każdy klient, który rozpoczyna komunikację z serwerem Apache przesyła do serwera żądanie dotyczące jakiegoś elementu zasobów. W odpowiedzi Apache albo dostarcza ten element, albo przesyła odpowiedź alternatywną, by wyjaśnić powody, dla których nie mógł spełnić żądania. Niejednokrotnie „elementem zasobów" może być strona WWW w języku HTML, umieszczona na lokalnym dysku — jest to jednak zaledwie jeden z wielu przypadków (w dodatku najprostszy). Oprócz strony WWW przedmiotem żądania może stać się plik graficzny, wynik działania skryptu, który wytwarza dane wyjściowe w języku HTML, aplet w języku Java i wiele innych.

Serwer Apache komunikuje się z klientami przy pomocy protokołu HTTP. Jest to protokół o charakterze *żądanie-odpowiedź*, co oznacza, że określa sposób formułowania żądań przez klienta oraz sposób, w jaki serwer powinien odpowiadać na te żądania. Plik wykonywalny serwera Apache wziął nazwę od protokołu; w systemach UNIX zazwyczaj działa jako `httpd` — nazwa ta oznacza demona protokołu HTTP.

Porównanie serwera Apache w systemach UNIX i Windows

Serwer Apache był w zamierzeniu stworzony do pracy w systemie UNIX. Obecnie można go w Internecie spotkać najczęściej w systemach Solaris, Linux i FreeBSD. Został też przeniesiony do Windows 95 i Windows NT i od tej chwili dokonał niemałych postępów względem uznanych serwerów Microsoft i Netscape. Niewątpliwie, biorąc pod uwagę zdobytą przez te firmy pozycję na rynku, ekspansja serwera Apache jest znaczącym osiągnięciem.

Z powodu pochodzenia z systemu UNIX, Apache 1.3 nigdy nie był równie dobry w Windows, więc korzystając z pojawienia się Apache w wersji 2.0, zdecydowano się zmienić budowę serwera. Jedną z najważniejszych zmian było oddzielenie logiki zależnej od platformy i włączenie jej do osobnego **modułu przetwarzania** (*MPM*, *Multi Processing Modul*). Ta innowacja wpłynęła korzystnie na funkcjonowanie Apache w systemach Windows — nowa implementacja serwera w Windows, korzystając z przeznaczonego dla tej platformy systemowej modułu MPM, działa znacznie szybciej i mniej zawodnie niż poprzednie.

Serwer Apache działa inaczej na platformie UNIX, niż w systemie Windows. W pierwszym przypadku Apache zwykle działa jako proces rozwidlany. Przy uruchomieniu Apache 1.3 tworzy (albo inaczej: rozwidla) kilka nowych procesów potomnych, służących do obsługi żądań. Każdy nowy proces utworzony w ten sposób jest całkowitą kopią macierzystego procesu serwera Apache. Serwer Apache 2.0 implementuje takie zachowanie dzięki modułowi MPM `prefork`, zaprojektowanemu dla zachowania zgodności z wersją 1.3.

System Windows nie ma nic podobnego do funkcji systemowej `fork` dostarczanej przez UNIX. Z tego względu nowa wersja serwera Apache została w znacznym stopniu zmieniona, by mogła skorzystać z implementacji wątków Windows. Teoretycznie jest to implementacja

prostsza i sprawniejsza, ponieważ wątki mogą dzielić zasoby, co zmniejsza zapotrzebowanie na pamięć. Ponadto wątki pozwalają systemowi operacyjnemu na inteligentne przełączanie zadań. Jednak implementacji wątków Apache w wersji 1.3 wykorzystywał warstwę emulacji wątków POSIX dla Windows. Z tego powodu Apache 1.3 nigdy nie działał tak dobrze, jakby mógł. Natomiast Apache 2.0, który korzysta wątków Windows, działa znacznie sprawniej.

Serwer Apache w wersji 2.0 obsługuje wątki na platformie UNIX za pomocą modułów MPM `worker` albo `perchild`, które dostarczają odmiennych modeli przetwarzania zależnie od wymagań. Nowa architektura serwera Apache, w połączeniu z walorami programowania wątków, spowodowała oczekiwane usprawnienie działania serwera. Równoczesne zmniejszenie różnic pomiędzy oprogramowaniem Apache dla systemów operacyjnych Windows i UNIX uprości tworzenie nowych aplikacji dla obu platform.

> Serwer Apache jest bardziej stabilny na platformach Windows NT, 2000 i XP niż Windows 9x i ME, dzięki lepszej implementacji wątków. W związku z tym jeśli planujemy uruchomienie serwera Apache w systemie Windows i chcemy zapewnić możliwie dużą niezawodność, powinniśmy wybrać wersję Windows NT, która dodatkowo umożliwia działanie Apache jako usługi systemowej.

Jeśli za najbardziej istotne cechy serwera uznamy niezawodność działania i bezpieczeństwo, to możemy brać pod uwagę tylko uruchomienie serwera w systemie UNIX. Taki wybór będzie korzystny nie tylko dla serwera Apache, ale i dla całego systemu. Ponadto nowsze wersje Apache w systemie UNIX osiągają stabilność szybciej niż w Windows. Aby jak najprędzej skorzystać z dobrodziejstw nowej wersji Apache, należy wybrać do jego instalacji platformę UNIX.

Apache może działać w wielu różnorodnych systemach operacyjnych; w większości z nich instalowany wprost z dystrybucji binarnej. Do tych systemów zaliczają się: OS/2, 680x0, Macintosh z procesorami PowerPC (zarówno sprzed wprowadzenia systemu MacOS X, jak i po jego wprowadzeniu), BeOS i Netware. Należałoby tu zwrócić baczniejszą uwagę na MacOS X, będący w rzeczywistości jedynie wariantem systemu UNIX. Apache w wersji 2.0 zapewnia standardowo moduły MPM dla systemów operacyjnych UNIX, Windows, OS/2, BeOS oraz Netware. Moduły MPM dla innych systemów operacyjnych są wciąż na etapie opracowywania. Dla uniknięcia niepotrzebnych dygresji, pominiemy omawianie modułów MPM dla innych systemów operacyjnych — zainteresowanych zaś odsyłamy na stronę WWW fundacji ASF, mieszczącą się pod adresem *http://www.apache.org/*.

Zagadnienia wyboru systemu operacyjnego będzie kontynuowane jeszcze nieco dalej — w podrozdziale „Wybór sprzętu komputerowego dla serwera".

Pliki konfiguracyjne i dyrektywy

Serwer Apache konfigurujemy za pomocą plików konfiguracyjnych, gdzie możemy zapisywać dyrektywy, które sterują jego zachowaniem. Apache obsługuje oszałamiającą liczbę dyrektyw, którą powiększa jeszcze każdy dołączony do serwera moduł.

To podejście daje administratorowi wszechstronne panowanie nad własnościami i bezpieczeństwem serwera. Konkurujące z serwerem Apache programy komercyjne nie zapewniają porównywalnej elastyczności; z tego względu Apache zyskuje nam nimi wyraźną przewagę.

Z drugiej strony, skomplikowana konfiguracja powoduje wydłużenie czasu potrzebnego na opanowanie serwera Apache. Wszystkie wysiłki są jednak warte nagrody, którą jest uzyskanie niemal całkowitej kontroli nad każdym aspektem działania serwera WWW.

Pewnym mankamentem wszechstronności serwera Apache jest brak zadowalającego rozwiązania, które pozwalałoby na jego pełną konfigurację za pomocą edytora z graficznym interfejsem użytkownika. W przypadku Apache, niektóre złożone ustawienia serwera muszą być przeprowadzane za pomocą ręcznej edycji plików konfiguracyjnych. Trwają już jednak obiecujące próby stworzenia narzędzia konfiguracyjnego na miarę plastyczności i siły serwera Apache. Któraś z nich spełni być może nasze oczekiwania. Więcej informacji na ten temat można znaleźć w rozdziale 2., w podrozdziale zatytułowanym „Graficzne narzędzia konfiguracyjne".

Spora część tej książki dotyczyć będzie (w różnym stopniu) dyrektyw konfiguracyjnych dla Apache. Najważniejsze z nich zostaną wprowadzone w rozdziale 4.; inne, bardziej zaawansowane będą pojawiać się w trakcie omawiania poszczególnych cech serwera. O konfiguracji będzie mowa także przy okazji innych problemów. W rozdziale 3. np. omówione zostanie tworzenie serwera Apache z kodu źródłowego, co daje sposobność użycia pewnych dodatkowych opcji konfiguracyjnych, niedostępnych w innych okolicznościach, a w rozdziale 10. zapoznamy się z kwestią bezpieczeństwa serwera WWW, rozpatrując ten problem pod kątem całego systemu komputerowego, a nie tylko z punktu widzenia Apache.

Moduły

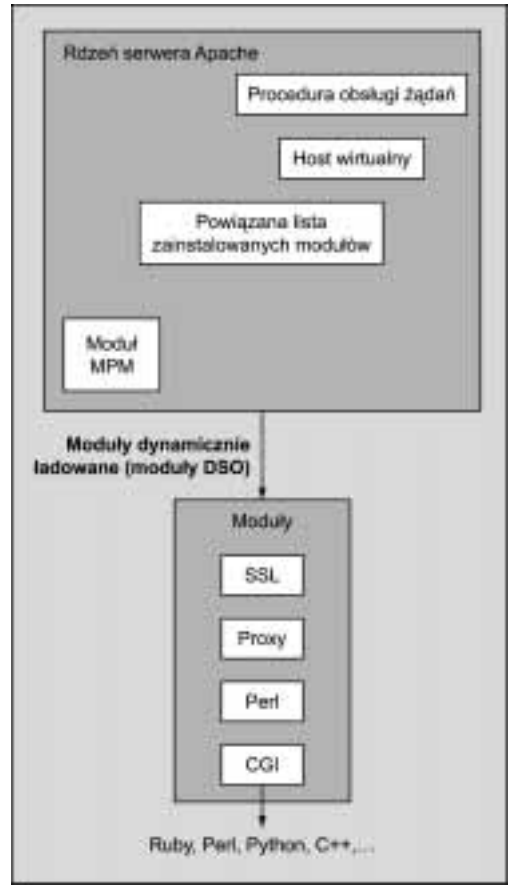
Jednym z najsilniejszych atutów serwera Apache jest jego struktura modularna. Główny program Apache ma tylko niezbędny zbiór cech. Pozostałych dostarczają moduły, albo wbudowane, albo ładowane dynamicznie przy uruchamianiu programu serwera. Apache w wersji 2.0 robi kolejny krok w tym kierunku i przenosi funkcje zależne od platformy do modułów MPM a monolityczne moduły (takie jak `mod_proxy` czy `mod_cache`) dzieli na moduły podstawowe i moduły implementacji. Dzięki temu można wybrać właściwy zestaw cech dostosowany do wymagań. Co więcej, rozwiązanie takie tworzy rozszerzalną architekturę, dogodną do stosowania w nowych typach serwerów pośredniczących i buforowych.

Konsekwencją struktury modularnej jest duża swoboda, jaką dysponuje administrator przy konfigurowaniu serwera Apache tworzonego z kodu źródłowego. Można dowolnie dołączać jedne, a rezygnować z innych modułów, usuwając niepotrzebne funkcje użytkowe. Wszystko to wpływa na zmniejszenie rozmiarów programu serwera, zmniejszenie wymaganej pamięci, mniejszą podatność na usterki konfiguracyjne — co w sumie zwiększa bezpieczeństwo serwera. Można także zwiększyć zestaw cech serwera Apache przez dodanie i uaktywnienie modułów, które standardowo nie są do niego dołączane.

Serwer Apache potrafi również dynamiczne dołączać moduły, dzięki czemu nie trzeba przebudowywać aplikacji, by dodać do niej jakieś nowe funkcje użytkowe. Wystarczy tylko zainstalować nowy moduł i ponownie uruchomić działający już serwer Apache, by moduł ten został dołączony.

W przypadku obsługi dynamicznego ładowania modułów serwer Apache zużywa nieco więcej pamięci niż w sytuacji, gdy takich modułów nie dołącza; ponadto wolniej się uruchamia, ponieważ musi załadować moduły z dysku. Jest to niewielka niedogodność, ale może okazać się kluczowa, gdy wymagana jest wysoka wydajność serwera.

Istnieje cały zestaw niezależnych modułów przeznaczonych dla serwera Apache. Niektóre z nich, jak choćby moduł `mod_fastcgi`, udostępniają dodatkowe wyspecjalizowane cechy, niezwykle zwiększające skuteczność serwera. Apache z modułem `mod_fastcgi` ma zdolność buforowania skryptów CGI, tak że mogą sprawniej odpowiadać na żądania użytkowników oraz powoduje, że skrypty te zużywają mniej zasobów systemu.



Inne moduły zwiększają efektywność i zakres możliwości serwera. Moduł `mod_perl` umożliwia całkowitą integrację interpretera języka Perl z serwerem Apache. Dzięki temu Apache może zrobić użytek z pełnego zestawu oprogramowania dostępnego dla Perla. Niektóre moduły kiedyś niezależne, zwłaszcza `mod_dav`, zostały włączone do dystrybucji Apache 2.0. Prawdopodobnie jednak najpoważniejszym nabytkiem nowej wersji Apache jest moduł `mod_ssl`, stanowiący podstawę dla obsługi protokołu SSL. Jest to moduł zintegrowany z serwerem Apache w jego wersji dystrybucyjnej 2.0. Eliminuje mnóstwo problemów (wśród nich konieczność wstawiania poprawek do kodu źródłowego Apache), z którymi do tej pory musieli borykać się administratorzy.

Wszechstronność, stabilność i wydajność Apache w połączeniu z otwartym dostępem do kodu źródłowego sprawiają, że jest on najczęściej stosowanym w Internecie oprogramowaniem serwera WWW.

Protokół HTTP

HTTP jest protokołem używanym przez wszystkie serwery WWW i programy klienta. Podczas gdy HTML jest językiem opisu stron WWW, protokół HTTP zajmuje się tym, jak programy klienta przedstawiają żądania, a serwery na nie odpowiadają.

Zazwyczaj HTTP działa niezauważalnie, niemniej zrozumienie podstawowych zasad tego działania może przydać się administratorowi serwera WWW w ocenie problemów i podejmowaniu decyzji w sprawie bezpieczeństwa. Wiele cech charakteryzujących serwer Apache wynika z właściwości protokołu HTTP. Biorąc to wszystko pod uwagę, warto poznać ten protokół, aby ułatwić sobie pracę z serwerem Apache.

Wersja HTTP/1.1 jest szczegółowo zdefiniowana w dokumencie RFC 2616, udostępnionym pod adresem *http://www.w3.org/Protocols/rfc2616/rfc2616.txt*.

Protokół HTTP jest protokołem bezstanowym (czyli nie utrzymującym żadnych informacji o stanie połączenia). Oznacza to, że dialog pomiędzy klientem WWW (na przykład przeglądarką) a serwerem składa się z przesłanego przez klienta żądania i wysłanej przez serwer odpowiedzi, wraz z wszelkimi operacjami pośrednimi, niezbędnymi do jej przygotowania.

Żądania i odpowiedzi HTTP

Pierwszy wiersz żądania HTTP tworzy metoda opisująca zamiar klienta, URI (*Uniform Resource Identifier*), zasobu do pobrania lub zmiany oraz wersja protokołu HTTP. W dalszej kolejności umieszczona jest pewna liczba nagłówków, które mogą wpływać na żądanie, na przykład uwarunkować je zależnie od pewnych kryteriów, lub, co jest wymagane w protokole HTTP/1.1, określić nazwę hosta. Po odbiorze żądania i towarzyszących mu nagłówków serwer wybiera kierunek działania i odpowiada we właściwy sposób. Typowe żądanie dostarczenia dokumentu HTML mogłoby wyglądać tak:

```
GET /index.html HTTP/1.1
Host: www.alpha-complex.com
```

> Używając polecenia `telnet` albo innego podobnego narzędzia możemy połączyć się z działającym serwerem, pisząc na przykład: `telnet localhost 80`, a po napisaniu obu wierszy wystarczy dwukrotnie nacisnąć klawisz *Return*, żeby wysłać żądanie i zobaczyć odpowiedź. W rozdziale 2. Czytelnik znajdzie więcej szczegółów na temat użycia polecenia `telnet`.

Zakończone powodzeniem żądanie zwraca kod stanu `200` oraz żądaną informację, poprzedzoną nagłówkami odpowiedzi serwera. Typowe nagłówki w odpowiedzi z serwera Apache wyglądają następująco:

```
HTTP/1.1 200 OK
Date: Mon, 09 Aug 2002 06:37:56 GMT
Server: Apache/2.0.34 (UNIX)
Last-Modified: Mon, 02 Aug 2002 06:31:48 GMT
ETag: "d456-68-248fdd00"
Accept-Ranges: bytes
Content-Length: 104
Content-Type: text/html; charset=ISO-8859-1
```

Pierwszy wiersz jest wierszem stanu; zawiera typ protokołu oraz kod oznaczający powodzenie. Następnie mamy kolejno: datę, kilka szczegółów na temat serwera oraz pozostałe nagłówki odpowiedzi. W zależności od serwera i żądania mamy do czynienia z różnymi typami nagłówków. Najważniejszy jest nagłówek `Content-Type`, który mówi klientowi co zrobić z odpowiedzią. Nagłówek `Content-Length` informuje klienta o tym, jaką długość ma treść odpowiedzi. Nagłówki `Date`, `ETag` i `Last-Modified` są używane do buforowania (*caching*).

W przypadku błędu w wierszu stanu zwracany jest kod błędu z opisem problemu, na przykład:

```
HTTP/1.1 404 Not Found
```

Serwer może w określonych okolicznościach zwrócić także inne kody. Taką okolicznością może być na przykład przekierowanie.

Metody HTTP

Metody informują serwer o rodzaju żądania. Niektóre wymieniono w zamieszczonej tu tabeli. Podane w niej przykłady, które mają zilustrować charakter żądania i odpowiedzi, zostały skrócone do niezbędnego minimum. Prawdziwy serwer Apache najprawdopodobniej wyśle znacznie większą liczbę nagłówków.

Metoda	Zadanie	Przykład
GET	Pobierz z serwera nagłówek i zasób.	Żądanie: GET /index.html HTTP/1.0
	(Pusty wiersz oddziela nagłówek od zasobu w odpowiedzi).	Odpowiedź: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 17:02:08 GMT Server: Apache/2.0.33 (UNIX) Content-Length: 1776 Content-Type: text/html; charset=ISO-8859-1 Connection: close <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <html> ... </html>
HEAD	Zwróć nagłówek, który byłby zwrócony przez GET, ale nie zwracaj zasobu.	Żądanie: HEAD /index.html HTTP/1.0
	Zauważmy, że zwracana jest długość zasobu lecz nie zawartość.	Odpowiedź: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 17:01:13 GMT Server: Apache/2.0.33 (UNIX) Content-Length: 1776 Content-Type: text/html; charset=ISO-8859-1 Connection: close

Metoda	Zadanie	Przykład

POST	Wyślij informację do serwera.	Żądanie:
		`POST /cgi-bin/search.cgi HTTP/1.0`
		`Content-Length: 46`
		
		`query=alpha+complex&casesens=false&cmd=submit`

	Odpowiedź serwera może zawierać potwierdzenie otrzymania wysłanej informacji. Serwer musi być odpowiednio skonfigurowany, by reagował na POST, uruchamiając przykładowo skrypt CGI.	Odpowiedź:
		`HTTP/1.1 201 CREATED`
		`Date: Mon, 09 Aug 2002 17:02:20 GMT`
		`Server: Apache/2.0.33 (UNIX)`
		`Content-Type: text/html; charset=ISO-8859-1`
		`Connection: close`
		
		`<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`
		`<HTML>`
		`...`
		`</HTML>`

W protokole HTTP/1.1 obsługiwane są również następujące polecenia:

Metoda	Zadanie	Przykład
OPTIONS	Zwróć wykaz dozwolonych przez serwer metod.	Żądanie: `OPTIONS * HTTP/1.1`

	Przykład odnosi się zwłaszcza do serwerów WebDAV, obsługujących dodatkowe metody, zdefiniowane w dokumencie RFC 2518.	Odpowiedź:
		`HTTP/1.1 200 OK`
		`Date: Mon, 09 Aug 2002 16:54:55 GMT`
		`Server: Apache/2.0.33 (UNIX)`
		`Allow: GET, HEAD, POST, OPTIONS, TRACE`
		`Content-Length: 0`
		`Content-Type: text/plain; charset=ISO-8859-1`

TRACE	Obserwuj żądanie, aby zobaczyć, co właściwie widzi serwer.	Żądanie:
		`TRACE * HTTP/1.1`
		`Host: www.alphacomplex.com`

	Zadaniem tej metody jest sprawdzenie, jak wygląda żądanie po przejściu przez serwery pośredniczące. Można ją skierować przez nagłówek Max-Forwards do pośredniczącego serwera, by uzyskać informację o pośredniczących serwerach.
	
	Więcej informacji o TRACE zawiera RFC 2616.
		Odpowiedź:
		`HTTP/1.1 200 OK`
		`Date: Mon, 09 Aug 2002 17:09:18 GMT`
		`Server: Apache/2.0.33 (UNIX)`
		`Content-Type: message/http; charset=ISO-8859-1`
		
		`TRACE  HTTP/1.1`
		`Host: www.alphacomplex.com`

Metoda	Zadanie	Przykład

DELETE	Usuń zasób z serwera.
	
	Serwer w zasadzie nie powinien na to zezwalać, więc próba użycia DELETE powinna spowodować odpowiedź jak w przykładzie. Wyjątkiem są serwery WebDAV, które implementują metodę DELETE.
		Żądanie:
		`DELETE /document.html HTTP/1.1`
		`Host: www.alphacomplex.com`
		
		Odpowiedź:
		`HTTP/1.1 405 Method Not Allowed`
		`Date: Mon, 09 Aug 2002 17:24:37 GMT`
		`Server: Apache/2.0.33 (UNIX) DAV/2`
		`Allow: GET, HEAD, OPTIONS, TRACE`
		`Content-Type: text/html; charset=ISO-8859-1`
		
		`<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`
		`<HTML><HEAD>`
		`<TITLE>405 Method Not Allowed</Title>`
		`</HEAD><BODY>`
		`<H1>Method Not Allowed</H1>`
		`The requested method DELETE is not allowed for the URL /document.html.<P>`
		`</BODY></HTML>`

PUT	Utwórz lub zmień plik na serwerze.
	
	Serwer w zasadzie nie powinien zezwalać na użycie PUT, ponieważ większość podobnych zadań spełnia POST. Metoda PUT, w odróżnieniu od POST, powoduje powstanie bezpośredniego związku między podanym w żądaniu PUT identyfikatorem URI i takim samym identyfikatorem URI w późniejszej metodzie GET, czego nie pociąga POST. Metodę PUT mogą implementować serwery WebDAV.
		Żądanie:
		`PUT /newfile.txt HTTP/1.1`
		`Host: www.alphacomplex.com`
		`Content-Type: text/plain`
		`Content-Length: 59`
		
		`To jest zawartosc pliku, ktory chcemy utworzyc na serwerze.`
		
		Odpowiedź:
		`HTTP/1.1 201 CREATED`
		`Date: Mon, 09 Aug 2002 17:30:12 GMT`
		`Server: Apache/2.0.33 (UNIX) DAV/2`
		`Content-Type: text/html; charset=ISO-8859-1`
		
		`<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`
		`<HTML>`
		`...`
		`</HTML>`

CONNECT	Metoda, która pozwala serwerom pośredniczącym na przejście w tryb tunelowania dla protokołów takich jak SSL.	Szczegółowe omówienie znajduje się w opisie dyrektywy AllowCONNECT w rozdziale 8.

Identyfikator URI

Identyfikator URI jest łańcuchem tekstowym, który identyfikuje element zasobów przez nazwę, lokalizację lub w innym formacie rozumianym przez serwer. Identyfikatory URI definiuje RFC 2396.

Identyfikator URI jest to zazwyczaj zwykły URL w postaci zrozumiałej dla przeglądarki. Najprostszy identyfikator URI ma postać /. Można podać tu podać każdy obowiązujący na serwerze identyfikator URI, na przykład:

/
/index.html
/centralcontrol/bugreport.htm:80
http://www.alpha-complex/images/ultraviolet/photos/outside.jpg

Jeśli metoda nie wymaga dostępu do określonego zasobu, można zastosować specjalny identyfikator URI w postaci gwiazdki (*), co pokazaliśmy to wcześniej w przykładzie `OPTIONS`. W takich przypadkach użycie obowiązującego identyfikatora URI nie jest błędne, lecz niepotrzebne.

Protokół HTTP

Protokół HTTP występuje w jednej z wersji:

- HTTP/0.9
- HTTP/1.0
- HTTP/1.1

W praktyce klient nigdy nie wysyła `HTTP/0.9`, gdyż argument protokół wprowadzono wraz z wersją `HTTP/1.0`, dla rozróżnienia żądań 1.0 od 0.9. Protokół `HTTP/0.9` jest domniemany, gdy klient nie wysyła protokołu, ale w ten sposób działają tylko `GET` i `POST`, gdyż pozostałe metody wprowadzono dopiero w wersji 1.0 protokołu HTTP.

Nagłówki HTTP

Nagłówki HTTP (znane także jako pola nagłówkowe HTTP) mogą towarzyszyć komunikatom HTTP przekazywanym w obu kierunkach pomiędzy klientem a serwerem. W praktyce może być wysłany dowolny nagłówek, o ile na obu końcach połączenia panuje zgoda co do jego sensu. Protokół HTTP definiuje jedynie pewien podzbiór nagłówków.

Oto klasyfikacja rozpoznawalnych przez protokół HTTP nagłówków:

- **Nagłówki żądania** (*request headers*) są wysyłane przez klienta dla uzupełnienia informacji o żądaniu lub modyfikacji jego charakteru. Na przykład nagłówek `Accept-Language` informuje serwer o językach, w jakich dokumenty mogą być zaakceptowane przez klienta. Nagłówek ten może być wykorzystany przez serwer Apache do negocjacji treści.
- **Nagłówki odpowiedzi** (*response headers*) są wysyłane przez serwer do klienta w odpowiedzi na jego żądanie. Standardowe nagłówki, zwykle wysyłane przez serwer Apache, to `Date` i `Connection`.
- **Nagłówki encji** (*entity headers*) są wysyłane zarówno przez serwer, jak i klienta. Wnoszą informację opisową (zwaną także meta-informacją) o treści komunikatu HTTP. Żądania HTTP mogą używać nagłówków encji tylko w metodach,

które zawierają treść komunikatu, tj. PUT i POST. Żądania zawierające treść komunikatu są zobowiązane do wysłania nagłówka Content-Length, określającego rozmiar treści. Serwery zamiast wysyłać nagłówek Content-Length, mogą wysłać nagłówek Transfer-Encoding.

Prócz nagłówków określających treść, do których należą także Content-Language, Content-Encoding i znany już Content-Type, warto wymienić jeszcze dwa inne: Expires i Last-Modified. Nagłówek Expires informuje przeglądarkę i serwery pośredniczące, jak długo dokument pozostanie aktualny. Nagłówek Last-Modified pozwala klientowi określić, czy dokument przechowywany przez niego w pamięci buforowej jest aktualny. Aby przekonać się o tym, że ten ostatni nagłówek jest istotnie przydatny, rozważmy serwer pośredniczący, który buforuje żądany dokument. Po otrzymaniu żądania od klienta, serwer pośredniczący najpierw wysyła żądanie HEAD do serwera WWW, a następnie szuka pola Last-Modified w otrzymanej odpowiedzi. Jeśli je znajdzie i nie jest ono nowsze od dokumentu w buforze, serwer pośredniczący nie żąda dokumentu z serwera WWW, lecz wysyła buforowaną wersję. Więcej na ten temat można będzie przeczytać w rozdziale 4., przy okazji opisu modułu mod_expires.

Pełen wykaz rozpoznawalnych przez HTTP nagłówków zamieszczono w dodatku H.

Praca w sieci oraz TCP/IP

Komputer może spełniać dobrze swoją rolę w izolacji, lecz po przyłączeniu do sieci jest na ogół bardziej użyteczny. Serwer WWW, aby być dostępny, musi łączyć się ze światem zewnętrznym.

Dla połączenia w sieć co najmniej dwóch komputerów potrzeba jakiegoś środka komunikacji. W biurze czy innej instytucji tym medium może być na przykład *Ethernet* z kartami sieciowymi w każdym komputerze i łączącymi kablami. Jednak sam sprzęt nie wystarcza. O ile można dać sobie radę, wysyłając dane po prostu przez połączenie szeregowe, to w przypadku komputerów w sieci z wieloma innymi komputerami konieczne jest określenie bardziej zaawansowanego protokołu, definiującego sposób przesyłu danych, ich dostarczania, odbierania i potwierdzania.

TCP/IP jest jednym z kilku protokołów używanych do komunikacji pomiędzy komputerami w sieci. Jest to protokół używany przede wszystkim w Internecie. Inne to *Token Ring* (nie działa przez *Ethernet*) oraz *SPX/IPX* (działa przez *Ethernet*). Na ogół stosuje się je w sieciach wewnętrznych (intranetach) większych przedsiębiorstw.

Definicje

TCP/IP to właściwie dwa protokoły, jeden nad drugim. Protokół niższego poziomu, IP, dotyczy tras dla danych przesyłanych pomiędzy nadawcą a odbiorcą. Rozdziela dane na pakiety, do których doczepia adres nadawcy i odbiorcy, jak na kopertach.

Obecnie dostępne są dwie wersje IP. Starsza i zarazem najczęściej stosowana wersja to *IPv4* (IP w wersji 4). IPv4 to protokół nadal powszechnie stosowany w sieci Internet, choć zaczął zdradzać oznaki starzenia się. Jego następcą jest protokół *IPv6*, który rozszerza zakres adresowania z 32 do 128 bitów, wprowadza obsługę IP dla komputerów przenośnych i obsługę *quality-of-service* oraz — co jest niemniej ważne — wprowadza opcjonalne uwierzytelnienie i szyfrowanie połączeń sieciowych. Ta część protokołu, odpowiedzialna za bezpieczeństwo pracy w sieci, jest tak obszerna, że opublikowano ją w odrębnej specyfikacji, znanej jako *IPSec*.

Protokół TCP pozostawia w gestii IP szczegóły przekazu danych z jednego miejsca w inne. Tworzy nad tym mechanizm ustanawiania połączeń, zapewnia właściwą kolejność odbieranych danych, obsługę przypadków utraty danych, błędów i przywracania połączeń. TCP określa protokół sterowania przepływem danych (*handshake*) pozwalający wykryć błąd sieci i definiuje swój własny zestaw informacji dla pakietów. Do zestawu tego należy numer porządkowy dodawany przez protokół TCP do pakietu wysyłanego przez IP. Numer ten pozwala zapewnić identyczną kolejność pakietów przy wysyłaniu i odbiorze.

TCP nie jest jedynym protokołem, który używa IP. Do rodziny protokołów TCP/IP należy również UDP (*User Datagram Protocol*). W odróżnieniu od TCP zorientowanego na połączenia i niezawodnego, UDP jest bezpołączeniowym protokołem bez gwarancji, używanym do mniej wymagającej transmisji i rozgłaszania, które na ogół dotyczy informacji mieszczących się w jednym pakiecie. Z uwagi na uproszczone działanie — UDP nie sprawdza, czy transmisja lub kolejność jest poprawna — stosowany jest on wtedy, gdy użycie TCP byłoby nieporęczne, na przykład w audycjach lub grach w Internecie. UDP jest też podstawą sieci *peer-to-peer* takich jak Gnutella (*http://www.gnutella.com*).

Rodzina TCP/IP obejmuje także protokół ICMP (*Internet Control Message Protocol*), który jest używany przez programy TCP/IP do przesyłania komunikatów dotyczących samego protokołu TCP/IP, takich jak niepowodzenie w połączeniu z danym hostem. ICMP jest przeznaczony raczej do użytku przez warstwy protokołu TCP/IP niskiego poziomu niż do wykorzystania z poziomu aplikacji użytkownika.

> Protokoły TCP, UDP, ICMP oraz IP są zdefiniowane w następujących dokumentach RFC: 768 (UDP), 791 (IP), 792 (ICMP), 793 (TCP). Dodatek A zawiera pełny wykaz przydatnych dokumentów RFC.

Pakiety i kapsułkowanie

Wspomniano już, że protokoły IP oraz TCP dodają informacje do pakietów danych, które potem są transmitowane pomiędzy hostami. Aby lepiej zrozumieć działanie TCP/IP, warto przyjrzeć się temu, co faktycznie robi IP i TCP.

Kiedy aplikacja przysyła jakiś blok danych — na przykład plik lub stronę HTML — protokół TCP rozdziela dane na pakiety przeznaczone do faktycznej transmisji. Standard Ethernet określa maksymalny rozmiar pakietu na 1500 bajtów; w starszych sieciach sprzęt może dodatkowo ograniczyć ten rozmiar do 576 bajtów. Po ustanowieniu połączenia protokół TCP/IP najpierw sprawdza, jak duży może być pakiet. Nawet jeśli lokalna sieć radzi sobie z pakietami

o rozmiarze 1500 bajtów, sieć występująca po drodze może nie akceptować tak dużych pakietów. Cała komunikacja musi odbywać się poprzez wymianę pakietów o maksymalnym akceptowalnym rozmiarze, chyba że sieć pośrednicząca potrafi dzielić pakiety na mniejsze.

Kiedy TCP wie jaki jest rozmiar pakietu, kapsułkuje każdy blok danych za pomocą nagłówka TCP, który zawiera numer porządkowy, adres źródłowy i docelowy oraz sumę kontrolną do wykrywania błędów. Nagłówek ten jest podobny do adresu na kopercie, zawierającej „list" w postaci pakietu danych.

Teraz protokół IP dodaje do pakietu TCP swój własny nagłówek, w którym zapisuje źródłowy i docelowy adres IP tak, że można określić trasę pakietu na kolejnych etapach. Protokół IP dodaje także typ protokołu, by zidentyfikować pakiet jako TCP, UDP, ICMP czy jakiś inny, oraz dodaje własną sumę kontrolną. Jeśli stosujemy IPv6, pakiet może zostać ponadto podpisany dla uwierzytelnienia nadawcy, a następnie zaszyfrowany do transmisji.

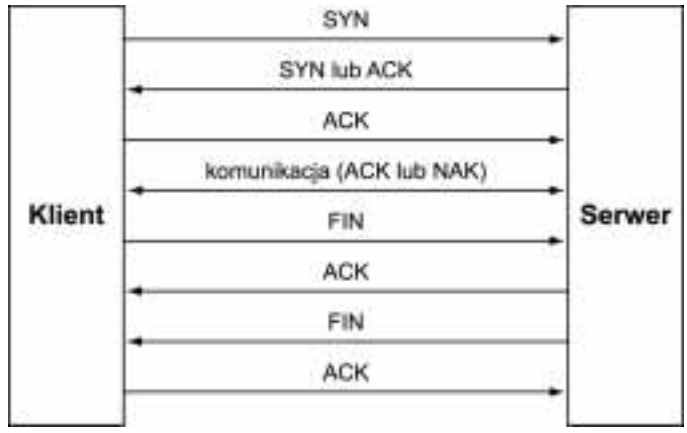
Ponadto, jeśli pakiet ma być przesłany przez sieć Ethernet, również Ethernet dodaje kolejny nagłówek, zawierający źródłowy i docelowy adres Ethernet, kod typu i jeszcze jedną sumę kontrolną. Konieczność umieszczenia nowego nagłówka wynika z tego, że Ethernet używa własnych adresów interfejsów sieciowych dla każdego etapu trasy, jaki pakiet pokonuje na drodze od hosta wysyłającego do odbierającego. Nagłówek nadany przez protokół IP zapisuje jedynie adresy IP początkowego i końcowego komputera. Każdy kolejny protokół działa na odległość mniejszą niż ten, który jest przez niego kapsułkowany, opisując coraz to krótsze skoki w podróży od źródła do celu.

Zarówno protokół IP, jak i TCP, dodają dwadzieścia bajtów informacji do pakietu danych, który w całości powinien zmieścić się w wyznaczonym przez Ethernet limicie 1500 bajtów; a zatem maksymalnie można umieścić w pakiecie TCP/IP tylko 1460 bajtów danych. Jeśli protokół IP nie działa przez Ethernet, a przez połączenie szeregowe, oczywiście nie ma konieczności ograniczania pakietu do 1500 bajtów. Inne protokoły mogą również nałożyć swoje własne ograniczenia na rozmiar pakietu.

Komunikaty ACK, NAK i inne

Większą część transmisji TCP stanowią opisane w poprzednim punkcie pakiety danych. Jednak ponieważ IP nie próbuje zapewnić doręczenia pakietu, TCP wymaga, aby odbiorca wysłał komunikat potwierdzenia odbioru, ACK. Komunikat taki informuje nadawcę, że dane dotarły do odbiorcy. Komunikaty ACK są zatem prawie tak częstym zjawiskiem, jak pakiety danych — w idealnej sieci powinno być ich dokładnie tyle samo. Jeśli z pakietem jest coś nie w porządku, protokół TCP wymaga, by odbiorca zamiast ACK wysłał komunikat o braku potwierdzenia, NAK.

Prócz danych i komunikatów ACK oraz NAK, protokół TCP definiuje także komunikaty synchronizacji, SYN, do ustanawiania połączeń i komunikaty FIN do rozłączania. Klient żąda połączenia, wysyłając komunikat SYN do serwera. Serwer ustanawia połączenie albo odmawia jego ustanowienia, odsyłając do klienta odpowiednio komunikat ACK lub NAK. Kiedy któraś ze stron chce zakończyć połączenie, wysyła do drugiej komunikat FIN, oznaczający koniec komunikacji.



Możliwe są zatem trzy scenariusze, jakie wysyłający dane host musi uwzględnić:

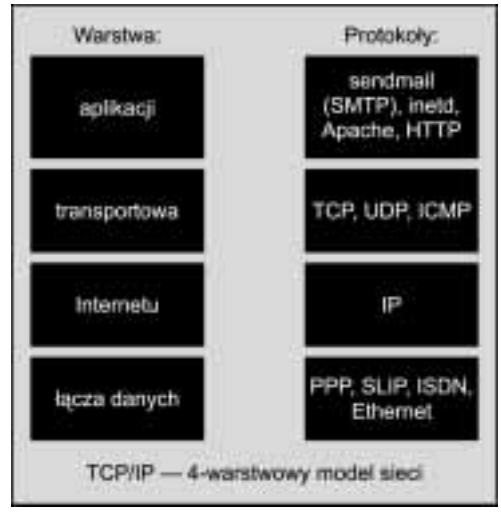
- Docelowy host odbiera pakiet. Jeśli był to oczekiwany pakiet lub żądanie ustanowienia nowego połączenia, wysyła komunikat ACK.
- Jeśli suma kontrolna pakietu nie zgadza się lub jego numer porządkowy jest niewłaściwy, host docelowy wysyła komunikat NAK, by poinformować hosta wysyłającego o konieczności ponownego wysłania pakietu.
- Host docelowy w ogóle niczego nie wysyła. W końcu TCP wnioskuje, że albo pakiet został zagubiony po drodze, albo odpowiedź musiała się gdzieś zapodziać i ponawia wysłanie pakietu.

Ataki z odmową usługi (*denial-of-service*) polegają na wyzyskaniu pewnych aspektów TCP/IP, żeby niepotrzebnie zająć serwer. Jednym z ataków DOS jest zalew SYN (*SYN flood*), gdy do serwera wysyłane są liczne pakiety SYN, ale przyjęcie żądanych połączeń nigdy nie jest potwierdzane przez klienta. Widzimy, że rozumienie TCP nie jest kwestią czysto akademicką. Radzenie sobie z takimi atakami jest jednym z tematów rozdziału 10.

Model sieci TCP/IP

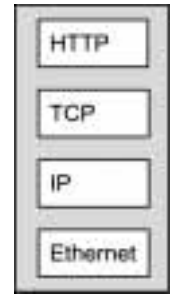
Protokoły TCP i IP tworzą dwie warstwy w hierarchii protokołów, która rozciąga się od aplikacji na samej górze aż do sprzętu na samym dole. Model sieci TCP/IP jest uproszczoną wersją siedmiowarstwowego modelu OSI. Model TCP/IP przypomina model sieci OSI, ale nie ma między nimi całkowitej zgodności. Chociaż często w różnych opracowaniach porównuje się model OSI do TCP/IP, te porównania są prawie zupełnie nieprzydatne, gdyż nic nie jest w pełni zgodne z modelem OSI. Cenniejsza jest wiedza o samym protokole TCP/IP.

TCP/IP jest hierarchią czterech poziomów sieci, w której nad sprzętem i pod aplikacją TCP i IP tworzą środek. Uproszczony diagram stosu wygląda tak:

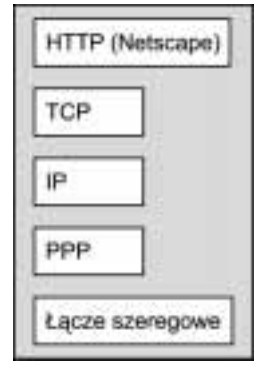


Warstwa łącza danych (*data link*) jest pokazana jako pojedynczy poziom, choć w praktyce zawiera wiele poziomów. Jednakże z punktu widzenia TCP/IP nie musimy się tym zajmować. Zobaczmy, jaką postać mogą mieć warstwy w przypadku typowej komunikacji serwera WWW z klientem.

- Po stronie serwera podłączonego do sieci Ethernet warstwy przedstawiają się następująco:



- Po stronie klienta, gdy użytkownik korzysta z dostępu przez łącze komutowane wyglądają one tak:



Dodatkowy protokół PPP pozwala działać IP nad szeregowym protokołem używanym między modemami, przez co warstwę łącza danych tworzą w tym przypadku dwie warstwy.

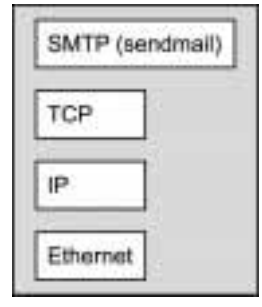
Kiedy użytkownik prosi o dostęp do strony WWW za pośrednictwem przeglądarki, przeglądarka generuje żądanie stosując protokół HTTP. Jest on transmitowany przez zainicjowane w TCP połączenie za pomocą protokołu IP, który wybiera trasę pakietu żądania do bramy przez łącze szeregowe, korzystając z protokołu PPP.

IP przekazuje pakiet trasą, na której może być wiele pośrednich serwerów. Adres zawarty w pakiecie mówi każdemu pośredniemu serwerowi dokąd skierować pakiet następnie.

W serwerze interfejs sieciowy widzi pakiet, którego adres IP wskazuje, że jest przeznaczony dla serwera. Serwer wyciąga ten pakiet z sieci i przekazuje go w górę do TCP, gdzie jest rozpoznawany jako żądanie połączenia, które zostaje potwierdzone. W chwilę później sieć widzi pakiet danych, który znowu jest przekazany w górę do TCP, gdzie jest identyfikowany jako pakiet właśnie ustanowionego połączenia. TCP potwierdza odbiór pakietu danych, oddziela informację kapsułkującą pakiet i przedstawia żądanie HTTP serwerowi Apache.

Apache przetwarza żądanie i odsyła odpowiedź do klienta. Znowu odpowiedź ta jest przekazywana w dół hierarchii, a następnie za pośrednictwem Internetu do klienta.

Gdybyśmy zarządzali systemem poczty na serwerze pocztowym w systemie UNIX, warstwy protokołów przedstawiałyby się następująco:



Jak widać, tylko protokół najwyższego poziomu i aplikacja jest inna niż dotąd. Całą resztę obsługuje TCP/IP.

Protokoły inne niż IP

Oprócz IP jest kilka innych protokołów, które działają bezpośrednio przez Ethernet i nie korzystają z IP. Na przykład do ustalania adresu Ethernet interfejsu sieciowego na podstawie adresu IP w sieciach Ethernet bywa używany ARP (*Address Resolution Protocol*). Konkurencyjne protokoły SPX/IPX także działają wprost przez Ethernet i nie wymagają IP. Sposób, w jaki Ethernet został zaprojektowany, pozwala współistnieć wszystkim tym protokołom.

Niewiele z tych protokołów można znaleźć w Internecie, bo większość nie potrafi przebyć drogi od źródła do celu z więcej niż jednym przeskokiem — to właśnie zapewnia protokół IP. Stąd protokoły, które tego wymagają, takie jak TCP lub UDP, budowane są nad IP zamiast niezależnie.

Adresy IP oraz klasy sieci

Każdy host w sieci TCP/IP powinien mieć niepowtarzalny adres IP, nadany przez administratorów sieci. Jeśli host ma komunikować się przez Internet, to w całym Internecie jego adres IP powinien być jedyny.

Adresy IPv4 są to 32-bitowe liczby, zapisane zwykle w czterech bajtach lub inaczej oktetach o wartościach od 0 do 255 rozgraniczonych kropkami, na przykład: `192.168.20.181`.

Adresy IPv6 są to 128-bitowe liczby, przedstawiane jako szesnastkowe bloki rozgraniczone dwukropkami, na przykład: `fe80::910:a4ff:aefe:9a8`. Widzimy, że nie ma tu dość cyfr, jak na 128-bitowy adres. Jest tak dlatego, bo miejsce zer w adresie zajmują dwukropki, tak że nie trzeba ich wypisywać jawnie. Zgodnie z intencją tylko częściowo mamy mieć wpływ na wybór tego numeru — jego część wyznacza adres Ethernet interfejsu sieciowego. Pozwala to automatycznie przydzielać adresy IPv6 i udostępnić sieć IP komputerom przenośnym, co było jednym z celów projektowych IPv6.

Cały zakres adresów IP jest podzielony na regiony, gdzie rezydują sieci różnych klas. Reszta Internetu traktuje adresy IP w danej klasie jako część jednej sieci i kierując pakiety do hostów w tej sieci spodziewa się jednego punktu kontaktowego, zwanego **bramą** (*gateway*).

Ponadto pewne adresy IP (pierwszy złożony z zer i ostatni złożony z liczb 255) uznaje się za specjalne w każdej klasie, więc adresów dla hostów jest nieco mniej niż można się spodziewać. Specjalnym adresom przyjrzymy się bliżej w następnym punkcie.

Przestrzeń adresowa protokołu IPv4, który ciągle obowiązuje w Internecie jako schemat adresowania, jest umownie podzielona na klasy sieci A, B oraz C. Oto charakterystyka tych klas:

- Nieliczne sieci klasy A zajmują zakres adresów, w których pierwsza liczba należy do przedziału od 1 do 126. Tylko pierwsza z czterech liczb jest ustalona w adresie sieci klasy A. Całkowita liczba adresów, jakie można nadać hostom w sieciach klasy A, to `16777214`.
- Sieci klasy B zajmują zakres od 128 do 191. Zarówno pierwsza, jak i druga liczba w adresie jest ustalona, co daje `16382` możliwych sieci klasy B, z których każda może mieć `65534` hostów.
- Sieci klasy C są najmniejsze; zajmują zakres od 192 do 223. Pierwsze trzy liczby w adresie są ustalone, co daje ponad dwa miliony możliwych sieci klasy C ale każda z tych sieci może zawierać tylko 254 hosty.
- Zakres od 224 do 254 jest zarezerwowany w specyfikacji protokołu TCP/IP.

Przestrzeń adresowa IPv6 podzielona jest podobnie, ale w szerszym zakresie: 6 oktetów (48 bitów) jest ustalone, a pozostałe 10 (80 bitów) jest przypisane do lokalnej sieci.

Specjalne adresy IP

Niektóre adresy IP są traktowane przez sieci TCP/IP specjalnie. W klasie sieci adres IP złożony z zer oznacza anonimowe źródło, gdy host nie potrafi określić źródłowego adresu IP — rzadki przypadek. Adres złożony z liczb 255 jest adresem rozgłaszania dla sieci (wszystkie

hosty w tej sieci mogą odebrać przekaz). Maska sieciowa nie jest adresem w ścisłym znaczeniu: określa które adresy w pewnym zakresie adresów IP są bezpośrednio połączone (tzn. leżą w tym samym segmencie sieci). Adresy, których różnica jest większa od maski sieciowej należą do różnych sieci i komunikują się za pośrednictwem bram oraz routerów.

W zależności od klasy sieci liczba zer w adresie anonimowym i liczb 255 w adresie rozgłaszania jest inna. Ilustrują to następujące trzy przykłady:

Klasa sieci	Adres anonimowy	Adres rozglaszania	Maska sieciowa
A	16.0.0.0	16.255.255.255	255.0.0.0
B	181.18.0.0	181.18.255.255	255.255.0.0
C	192.168.32.0	192.168.32.255	255.255.255.0

Ponieważ rozgłaszanie jest bezpołączeniowe — nadawca wysyła dane do każdego hosta, który może je odebrać — odbywa się za pomocą protokołu UDP. Protokół IPv6 różni się pod tym względem od IPv4. IPv6 nie obsługuje rozgłaszania (*broadcasting*). Zamiast tego stosuje **rozsyłanie grupowe** (*multicasting*). Dla uproszczenia pominiemy tę różnicę i będziemy się trzymać IPv4.

Są też i inne specjalne zakresy adresów IP, które są odmiennie traktowane przez sprzęt sieciowy, taki jak routery. Adresy IP z tych kilku zakresów traktuje się jako prywatne — pakiety nadawane na te adresy nie są nigdy transmitowane przez routery poza sieć lokalną. Z tego powodu adresy prywatne nadają się do testowania sieci lub do komunikacji w sieci intranet, która nie łączy się bezpośrednio z Internetem. Oto pełna lista takich adresów:

Klasa sieci	Prywatne sieci
A	10.0.0.0
B	**od** 172.16.0.0 **do** 172.31.0.0
C	**od** 192.168.0.0 **do** 192.168.255.0

Inny specjalny adres to adres pętli zwrotnej, 127.0.0.1, który odnosi się do lokalnego hosta (często nazwanego całkiem trafnie: localhost). Adres ten jest stosowany do komunikacji z serwerami uruchomionymi na lokalnym komputerze.

Inne adresy w sieci 127 są używane przez serwery poczty elektronicznej do identyfikacji otwartych przekaźników (*relays*) oraz innych niepożądanych źródeł poczty. Usługi takie jak MAPS, ORDB, ORBZ oraz Spews mają własne serwery DNS, które zwracają jakiś adres z sieci 127, gdy adres IP nadawcy pochodzi z czarnej listy. Taki adres nie jest poprawny, więc pozwala uzyskać efektywne rozstrzygnięcie tak(nie) od serwera DNS. Nie jest to standardowe użycie adresowania TCP/IP, ale — co trzeba przyznać — przynosi efekty.

Maski sieciowe i wybór tras

Adresy IP składają się z dwóch części: adresu sieci i adresu lokalnego hosta. Klasy sieci A, B i C odpowiadają sieciom z całkowitą liczbą oktetów ale możemy rozdzielić adres sieci i adres lokalny w dowolnym punkcie, używając arytmetyki dwójkowej.

Maska sieciowa wygląda jak adres IP, ale nim nie jest. W logicznym iloczynie z adresem IP daje adres sieci — `181.18.0.0` w przykładzie klasy B powyżej. Jeśli dwa adresy w iloczynie z maską dają ten sam adres sieci, to należą do tej samej sieci; w przeciwnym razie należą do różnych sieci. Maski sieciowe IPv6 są podobne do swoich odpowiedników IPv4, tylko dłuższe.

Maska sieciowa jest podstawowym atrybutem interfejsu sieciowego, tak jak adres IP. Używając maski sieciowej serwer może określić, czy docelowy adres IP jest lokalny i można się z nim połączyć bezpośrednio, czy trzeba łączyć się z nim pośrednio poprzez router. Dla przykładu rozważmy trzy adresy IP:

Adres IP	Nazwa hosta
`192.168.1.1`	*host_a*
`192.168.1.2`	*host_b*
`192.168.2.1`	*host_c*

Jeśli zdefiniujemy maskę sieciową `255.255.255.0` dla interfejsów sieciowych każdego hosta, pierwsze dwa hosty o nazwach *host_a* i *host_b* należą do tej samej sieci. Jeśli host *host_a* wysyła pakiet, TCP/IP spróbuje go przesłać bezpośrednio do hosta *host_b*. Jednakże, jeśli *host_a* próbuje wysłać pakiet do hosta *host_c*, nie może zrobić tego bezpośrednio, bo według maski sieciowej `192.168.1` i `192.168.2` są różnymi sieciami. Wyśle go więc do bramy. Konfiguracja każdego hosta zawiera informację o adresie IP co najmniej jednej bramy, do której dany host ma wysyłać pakiety, których sam nie może doręczyć.

Jeśli jednak zdefiniujemy maskę sieciową `255.255.0.0`, wszystkie trzy hosty będą należeć do tej samej sieci. W takim przypadku *host_a* będzie mógł wysyłać dane bezpośrednio do hosta *c*, oczywiście przy założeniu, że oba komputery są połączone z tą samą siecią fizyczną.

Protokół IP jest odpowiedzialny za to, by zaadresowany do jakiegoś hosta pakiet został doręczony do tego hosta. Podział przestrzeni adresowej na logiczne sieci znacznie ułatwia zadanie odnalezienia jakiegoś hosta. Wysyłający host nie musi znać wszystkich adresów w Internecie — wystarczy, że ze znanego wykazu bram wybiera adres, który jest następnym logicznym krokiem na drodze do celu. Oznaczenie następnego przystanku odbiera niższy protokół (na przykład Ethernet), który przekazuje tam pakiet. W przypadku protokołu Ethernet, tym oznaczeniem jest adres Ethernet bramy w sieci lokalnej. Brama przeprowadza tę samą procedurę, używając własnej listy bram i tak dalej, aż pakiet osiągnie ostatnią bramę i miejsce przeznaczenia.

> Przykład masek sieciowych w działaniu daje wydruk z polecenia `ifconfig`, zamieszczony dalej w tym rozdziale. Pokazuje on dwa lokalne adresy i zewnętrzny interfejs Ethernet przydzielone przez maskę sieciową do odrębnych sieci.

Odszukiwanie usług — dobrze znane porty

Klient kontaktuje się z serwerem, ponieważ na ogół chce skorzystać z pewnej usługi, takiej jak poczta elektroniczna lub FTP. Protokół TCP rozróżnia usługi używając definicji portu, dzięki czemu pojedynczy interfejs sieciowy może dostarczać wielu różnych usług. Kiedy klient zgłasza żądanie połączenia z serwerem, podaje nie tylko jego adres, czego wymaga IP, ale i numer portu.

Serwery HTTP takie jak Apache domyślnie obsługują port 80, który jest standardowym portem dla protokołu HTTP. Kiedy nadchodzi żądanie połączenia dla portu 80, system operacyjny wie że tego portu pilnuje serwer Apache i kieruje żądanie do niego. Każda standardowa usługa sieciowa i każdy protokół mają numer portu, z którym klienci mogą się łączyć dla uzyskania usługi HTTP, FTP, `telnet` czy innej.

Numery portów dla systemu UNIX definiuje standardowo plik o nazwie */etc/services*, zawierający listę wszystkich przydzielonych numerów. Podobny plik dla systemu Windows ma nazwę *Services* i jest położony w katalog instalowanym przez Windows *C:\WINNT\system32\drivers\etc*. W rzeczywistości system operacyjny oraz rozmaite demony odpowiedzialne za świadczenie usług już wiedzą, jakich portów mają używać — plik */etc/services* służy innym aplikacjom, by mogły odwoływać się do danej usługi przez nazwę, a nie numer portu. Plik ten określa ponadto, którego z protokołów — TCP lub UDP — używa dana usługa. Wiele usług obsługuje zarówno połączenia TCP, jak i UDP. Oto lista niektórych najczęściej stosowanych numerów portów z typowego pliku */etc/services*:

```
ftp         21/tcp                      # File Transfer Protocol
finger      79/tcp                      # Finger Daemon
www         80/tcp      http            # WorldWideWeb HTTP
www         80/udp                      # HyperText Transfer Protocol
pop-2       109/tcp     postoffice      # Post Office Protocol
pop-2       109/udp                     # Version 2
pop-3       110/tcp                     # Post Office Protocol
pop-3       110/udp                     # Version 3
nntp        119/tcp     readnews untp   # USENET News Transfer Protocol
ntp         123/tcp                     # Network Time Protocol
ntp         123/udp                     #
imap2       143/tcp     imap            # Interactive Mail Access
imap2       143/udp     imap            # Protocol V2
snmp        161/udp                     # Simple Net Management Protocol
imap3       220/tcp                     # Interactive Mail Access
imap3       220/udp                     # Protocol V3
https       443/tcp                     # Secure HTTP
https       443/udp                     # Secure HTTP
uucp        540/tcp     uucpd           # UNIX to UNIX Copy
```

Zwróćmy uwagę na port HTTP numer 80 i HTTPS numer 443 — na obu są akceptowane zarówno połączenia TCP, jak i UDP. Ich obsługa na danym porcie zależy od programu, który go obsługuje. Obecność usługi w wykazie nie oznacza, że serwer odpowie na próbę połączenia z nią. W rzeczywistości jest wiele powodów, dla których lepiej nie odpowiadać na żądania pewnych usług — np. usługi `telnet`, `ftp`, `snmp`, `pop-3` i `finger` są całkowicie nie związane ze stronami WWW i mogą być wykorzystane do osłabienia bezpieczeństwa serwera.

W systemach UNIX numery portów poniżej 1024 są zarezerwowane dla usług systemowych i programy uruchamiane przez nieuprzywilejowanych użytkowników nie mogą ich używać. Aby na porcie 80, standardowym porcie HTTP, mógł działać serwer Apache, musi go uruchomić *root*, albo system operacyjny, gdy rozpoczyna pracę. Użytkownicy nieuprzywilejowani mogą uruchamiać serwer Apache, o ile skonfigurują go tak, że używa numeru portu 1024 lub wyższego. W Windows nie ma takiego zabezpieczenia.

Sieciowy super serwer — inetd

Nie każda usługa, którą udostępnia host, jest obsługiwana przez nieustannie działającego demona. Byłoby to dużym marnotrawstwem zasobów. Z tego powodu system UNIX uruchamia wiele spośród swoich usług za pośrednictwem **demona Internetu** (`inetd`). Jest to nadrzędny serwer, który nasłuchuje na wielu różnych portach i uruchamia program do obsługi połączeń z chwilą, gdy otrzyma żądanie ustanowienia połączenia na którymś z tych portów.

Jedną z uruchamianych w ten sposób usług jest FTP; zwykle jest ona skojarzona z portem 21. W odróżnieniu od serwera Apache, który zazwyczaj działa samodzielnie i przejawia się jako kilka procesów `httpd`, w normalnych warunkach nie ma działającego procesu `ftpd`. Jednakże demon `inetd` sprawdza port 21, a kiedy otrzyma żądanie połączenia w protokole TCP, uruchamia kopię demona `ftpd` dla obsługi tego połączenia. Z chwilą uruchomienia demon `ftpd` negocjuje swoje własne prywatne połączenie z klientem, pozwalając superserwerowi zająć się ponownie nasłuchem. Z chwilą zakończenia komunikacji — w przypadku protokołu FTP, gdy transfer pliku został zakończony lub przerwany — demon `ftpd` kończy działanie.

Apache 1.3 posiada dyrektywę konfiguracji `ServerType`, która pozwala uruchomić ten serwer jako usługę, albo przez demona `inetd` tak, jak w przypadku FTP. W tej drugiej konfiguracji nie ma żadnego procesu `httpd`, chyba że demon `inetd` otrzyma żądanie połączenia dla portu 80 (czy innego, dla którego skonfigurowano `inetd` do uruchomiania Apache). Wtedy demon `inetd` uruchamia demona `httpd`, któremu przekazuje nadchodzące połączenie. Ponieważ Apache jest uruchamiany niezależnie dla każdego indywidualnego połączenia klienta, indywidualna kopia serwera działa tak długo, jak jest to niezbędne dla wypełnienia odebranego żądania. Jest to bardzo nieefektywna metoda pracy Apache, więc niemal we wszystkich konfiguracjach działa on jako samodzielna usługa. W Apache 2.0 usunięto tę dyrektywę.

Demon `inetd` może powodować dość istotne problemy. Jako główny demon koordynujący działanie demonów usług sieciowych, stanowi jedną z najpoważniejszych przyczyn naruszeń bezpieczeństwa sieci. Sam demon nie stwarza zagrożeń, ale realizuje usługi takie jak `telnet`, które nie są bezpieczne. Mając to na względzie, wielu administratorów WWW decyduje się na jego wyłączenie, tym bardziej, że żadna z usług, którymi zarządza, nie jest konieczna dla serwera WWW. Nowsze dystrybucje systemów UNIX są dostarczane z ulepszonym demonem o nazwie `xinetd`, który ma wbudowane dodatkowe środki bezpieczeństwa. Nadal jednak nie widać istotnych powodów, by uaktywniać tego demona. W rozdziale 10. można znaleźć dalsze informacje na ten temat.

Przyszłość — protokół IPv6

Obecnie używany protokół IP, a zatem IPv4, używa czterech 8-bitowych liczb do utworzenia adresu IP. Daje to 2^{32} możliwych adresów. Nawet biorąc pod uwagę anonimowe adresy oraz adresy rozgłaszania jest to teoretycznie wystarczająca liczba, by przydzielić jeden, niepowtarzalny adres niemal każdej osobie na planecie, a z pewnością każdemu właścicielowi komputera. Niestety, z powodu podziału dostępnych adresów IP na klasy sieci A, B i C liczba adresów do wykorzystania jest już na wyczerpaniu.

W odpowiedzi na ograniczenia protokołu IP w wersji 4 (IPv4) powstała wersja 6 tego protokołu — IPv6. Nowa wersja przewiduje zastosowanie 128-bitowych adresów zamiast 32-bitowych używanych obecnie. Zmienia się także konwencja zapisu adresów. W wersji IPv4 adresy zwykle są zapisywane jako oddzielone kropkami cztery liczby dziesiętne. Adresy IPv6 mają postać ośmiu czterocyfrowych liczb szesnastkowych rozdzielonych dwukropkami. Dla skrócenia zapisu adresu, można pominąć wiodące zera — wpisuje się wtedy podwójny dwukropek dla oznaczenia miejsca, w którym je pominięto. Na przykład, adres IPv6 zapisany w skrócie jako `fe80::910:a4ff:aefe:9a8` ma następującą pełną postać `fe80:0910:0000:0000:0000:a4ff:aefe:09a8`. Protokół IPv6 pozwala wygenerować astronomiczną liczbę 2^{128} możliwych adresów IP.

IPv6 wprowadza ponadto obsługę kilku innych ważnych właściwości. Jedną z nich jest informacja o jakości usługi, która pozwala na ustalanie priorytetów w transmisji danych. Dzięki temu serwery mogą obsługiwać z wyższym priorytetem ruch HTTP niż pocztę elektroniczną. Inną nową cechą protokołu IPv6 jest uwierzytelnianie i szyfrowanie opisane przez IPSec — specyfikację zabezpieczeń wbudowaną w protokół IPv6.

> IPSec jest, w najkrótszym ujęciu, zamiennikiem SSL — ma jednak znacznie szerszy zakres możliwości, który obejmuje uwierzytelnianie i zabezpieczanie dostarczania indywidualnych pakietów informacji. IPSec jest podstawą współczesnych wirtualnych sieci prywatnych VPN. Może też stać się przedmiotem zainteresowania ze strony firm, które poszukują sposobu bezpiecznego rozszerzenia własnych sieci intranet na swoje odległe przedstawicielstwa i dla potrzeb komputerów przenośnych.

Obsługa IPv6 jest obecnie powszechnie dostępna dla większości platform, chociaż najdłużej ma ją Linux i BSD a platformy komercyjne od niedawna. Serwer Apache 2.0 obsługuje adresy IPv6 we wszystkich dyrektywach, które dotyczą sieci. Najważniejsze z nich to `Listen`, `VirtualHost`, `allow` i `deny`. Jednak mimo korzyści, jakie oferuje IPv6, rzeczywista implementacja sieci IPv6 nadal postępuje powoli.

Szerokie zastosowanie protokołu IPv6 nastąpi dopiero wtedy, gdy liczba serwerów wspierających ten protokół będzie odpowiednio duża; warto zatem rozważyć dodanie protokołu IPv6 do konfiguracji Apache. Trzeba zachęcać dostawcę internetowego do dodania obsługi tego protokołu na serwerze, z którego korzystamy. Jeśli dostawca nie może obsługiwać IPv6, spróbujmy go do tego przekonać albo zmieńmy dostawcę.

IPv6 jest w istocie odrębną siecią, która działa niezależnie obok sieci IPv4. Podstawowa sieć, która tworzy IPv6 w okresie powstawania i wprowadzania nowego protokołu, znana jest pod nazwą sieci szkieletowej IPv6 (**6bone**). Punkty dostępu do tej sieci znajdują się w większości krajów. Są trzy sposoby zdobycia adresu IPv6 i włączenia się do sieci IPv6. Można:

- Zdobyć adres 6bone przydzielony za pośrednictwem dostawcy usług internetowych, który jest już uczestnikiem sieci 6bone.
- Zdobyć adres producenta IPv6 (*production address*) od dostawcy usług internetowych wraz z identyfikatorem sieciowym najwyższego poziomu. Nadawaniem tych adresów zajmuje się *International Regional Internet Registry*.
- Zastosować tunelowanie IPv6 w IPv4 dla połączenia lokalnego adresu IPv4 z zewnętrznym adresem IPv6. Adresy w zakresie tunelowania rozpoczynają się od 2002, potem następuje adres IPv4 routera w sieci lokalnej. Pozostałe bity tworzą lokalną część adresu IPv6 i są przyznawane przez dostawcę usług internetowych.

Więcej informacji na temat 6bone oraz IPv6, a w tym szczegółowe instrukcje rutynowego korzystania z sieci IPv6, można zdobyć zaglądając na witrynę *http://www.6bone.net/*. Godna polecenia jest zwłaszcza strona WWW wyjaśniająca w jaki sposób można przystąpić do 6bone.

Narzędzia sieciowe

Administrowanie siecią jest złożonym i wymagającym zajęciem, którego nie będziemy tutaj omawiać. Niektóre aspekty administrowania pod kątem wydajności i zabezpieczeń są przedstawione odpowiednio w rozdziałach 8. i 10. Niezależnie od tego, administrator serwera WWW może niekiedy skorzystać z kilku przydatnych narzędzi do rozwiązywania problemów, jakich przysparza serwer. Spośród wszystkich systemów operacyjnych zasadniczo najlepiej wyposażony w takie narzędzia jest UNIX, ponieważ ewoluował w ścisłym powiązaniu z Internetem i jest najczęściej stosowany do implementowania systemów internetowych.

System UNIX zawiera wiele programów narzędziowych, które mogą okazać się przydatne dla administratora sieci. Dla uzyskania wartościowych informacji mogą posłużyć administratorowi następujące narzędzia:

ifconfig

Program *ifconfig* jest obecnym w każdym systemie UNIX, standardowym narzędziem do konfiguracji interfejsu sieciowego (*if* pochodzi od *interface*). Narzędzie to może być wykorzystane do wyświetlenia bieżącej konfiguracji interfejsu sieciowego. Uprzywilejowany użytkownik może użyć tego programu do zmiany dowolnego parametru interfejsu sieciowego, takiego jak karta Ethernet, szeregowe łącze PPP czy też interfejs *loopback*. Aby wyświetlić, na przykład, konfigurację wszystkich interfejsów sieciowych hosta, należy wydać następujące polecenie:

```
$ /sbin/ifconfig -a
```

W systemie Windows analogiczne polecenie ma nazwę `ipconfig`:

```
> ipconfig /all
```

Host z jedną kartą Ethernet może zgłosić konfigurację podobną do następującej, pokazując dwa interfejsy:

```
eth0      Link encap:Ethernet  HWaddr 00:10:A4:FE:09:68
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.128
          inet6 addr: fe80::910:a4ff:aefe:9a8/10 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:9109 (8.8 Kb)  TX bytes:5658 (5.5 Kb)

lo        Link encap: Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1540 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:1540 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:231276 (225.8 Kb)  TX bytes:231276 (225.8 Kb)

lo:1      Link encap:Local Loopback
          inet addr:192.168.1.131  mask:255.255.255.128
          UP LOOPBACK RUNNING  MTU:16436  Metric:1

lo:2      Link encap:Local Loopback
          inet addr:192.168.1.132  mask:255.255.255.128
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

Pierwszym interfejsem jest karta Ethernet z unikatowym adresem Ethernet, zapisanym na stałe przez producenta oraz adresem IP i maską sieciową, które można konfigurować. Ten interfejs należy do serwera, który obsługuje protokół IPv6, więc ma zarówno adres IPv4, jak i IPv6. Adres IPv4 ma maskę sieciową, która przydziela go do sieci klasy C, oraz adres rozgłaszania, który jest kombinacją adresu IP i maski sieciowej. Polecenie `ifconfig` informuje nas, że interfejs działa sprawnie i jest zdolny do rozgłaszania, jak również poda aktywności interfejsu.

> MTU (*Maximum Transmission Unit*) wynosi 1500 — maksimum w sieci Ethernet.

Drugim interfejsem z podanego przykładu jest *loopback*. Urządzenie takie nie jest uzależnione od żadnego faktycznego sprzętu — nie ma zatem ani adresu Ethernet, ani adresu rozgłaszania. Ponieważ ograniczenie rozmiaru pakietu w sieci Ethernet nie stosuje się do interfejsu pętli zwrotnej, urządzenie radzi sobie z pakietami o rozmiarze do 16436 bajtów. Wszystkie dane wysłane przez interfejs muszą do niego wrócić — ilość danych odebranych równa się dokładnie ilości danych wysłanych.

Trzeci i czwarty interfejs są urządzeniami pozwalającymi na zdefiniowanie zastępczych adresów IP (*IP aliases*). Ta właściwość jest dostępna w niektórych współczesnych systemach operacyjnych. Pozwala na przypisanie do tego samego interfejsu kilku adresów IP, dzięki czemu powstają wirtualne interfejsy. Tu zastępcze adresy IP dotyczą pętli zwrotnej, jednak moglibyśmy określić je również dla karty Ethernet i odpowiadać na żądania dotyczące adresów widocznych na zewnątrz.

Należy zauważyć, że adres zastępczy nie musi być w żaden sposób związany z adresem podstawowego interfejsu. W podanym przykładzie interfejsy mają adresy zastępcze tej samej klasy C, co adres interfejsu Ethernet. Jednak ponieważ z definicji te adresy należą do różnych sieci, maska jest ustawiona tak, że wartość ostatniego oktetu z przedziału 0 – 127 jest różna od 128 – 255. Ponieważ interfejsami o adresach zastępczych są odpowiednio 131 i 132, należą do innej sieci niż Ethernet, którego ostatnim oktetem jest 1.

Oczywiście argumenty wiersza poleceń `ifconfig` i format wyników mogą być różne w różnych systemach operacyjnych. Warto wywołać `man ifconfig` i otworzyć stronę podręcznikową `ifconfig` w swoim systemie

netstat

Program *netstat* jest standardowym narzędziem systemu UNIX do monitorowania sieci w tym systemie. Narzędzie to potrafi wydobyć wiele różnych informacji na temat wszystkich lub

tylko jednego wybranego interfejsu sieciowego. Krótkie zestawienie wybranych argumentów `netstat` daje pojęcie o jego możliwościach (mogą wystąpić drobne różnice pomiędzy implementacjami dla różnych systemów operacyjnych):

Argument	Efekt
(brak argumentu)	Wyświetla otwarte połączenia (gniazda)
`-a`	Dodatkowo pokazuje gniazda nasłuchujące i te, które nie nasłuchują
`-c`	Odświeża wybraną tabelę w sposób ciągły
`-i`	Wyświetla interfejsy sieciowe
`-n`	Wyświetla adresy IP, nie ustala nazw
`-r`	Wyświetla trasy
`-s`	Wyświetla statystykę
`-v`	Dostarcza informacji w trybie opisowym

Polecenie `netstat` obsługuje znacznie więcej argumentów, zwłaszcza dla domyślnej tablicy (otwartych połączeń sieciowych) — szczegóły podaje strona podręcznikowa *http://snowhite.cis.uoguelph.ca/course_info/27420/netstat.html*.

snoop i tcpdump

Oba te programy umożliwiają administratorowi badanie pakietów wysyłanych do sieci. Program *snoop* jest dostępny w systemie *Solaris*, zaś `tcpdump` jest podobnym nieodpłatnym narzędziem na platformach Linux i BSD (choć nadaje się dla każdej platformy, gdzie można je skompilować, bo kod źródłowy jest dostępny bez ograniczeń).

Oba narzędzia pozwalają na badanie pakietów, gdy pojawiają się w sieci. Różnorakie opcje pozwalają filtrować pakiety pod kątem źródłowego adresu IP i portu oraz docelowego adresu IP i portu, a także protokołu, typu komunikatu i tak dalej. Na przykład komunikację Apache można monitorować na porcie 80 w zakresie pakietów danych.

Zauważmy, że nie jest konieczne zarejestrowanie się w systemie serwera, by przeprowadzać monitorowanie pakietów. Do wykonania tego zadania wystarczy dowolny komputer przyłączony do tej samej sieci co serwer. Zwykle jednak, ze względów bezpieczeństwa, system UNIX wymaga, by użytkownik miał uprawnienia *root* do podsłuchiwania w tej sieci.

ping

Polecenie `ping` to najprostsze i najpożyteczniejsze narzędzie sieciowe. Wysyła komunikat ICMP do zdalnego hosta — określonego poprzez nazwę lub adres IP — dla ustalenia, czy jest on obecny i osiągalny. Polecenie zgłasza czas potrzebny na przebycie przez ten komunikat drogi tam i z powrotem. Większość wersji programu `ping` pozwala na testowanie połączeń ze zdalnym hostem w regularnych odstępach czasu, co przydaje się to do zapobiegania wygasaniu i rozłączaniu połączeń.

spray

Narzędzie to, którego nazwa może różnić się zależnie od systemu, jest odmianą polecenia `ping`. Powoduje zalewanie serwera docelowego pakietami *ping* dla sprawdzenia zdolności do ich obsługi przez sieć i serwer. Im wyższy procent pakietów, które docierają do celu, tym lepsza sieć. Nie należy nadużywać `spray` w sieci, która obsługuje rzeczywisty ruch.

traceroute

Polecenie `traceroute` jest użyteczne w diagnozowaniu problemów z nawiązywaniem połączeń, na przykład wtedy, gdy `ping` bez powodzenia próbuje dotrzeć do zdalnego serwera. Narzędzie `traceroute` używa protokołu ICMP dla zdobycia informacji o każdym pośrednim etapie trasy od hosta do celu. Może zwracać więcej niż ponad dwadzieścia wierszy w przypadku tras w Internecie.

Polecenie `traceroute` jest szczególnie przydatne w diagnozowaniu problemów związanych z nieudanymi połączeniami — czasem jest w stanie wskazać miejsce na trasie połączenia, w którym występują usterki. Może też pomóc w wykryciu nieprawidłowo skonfigurowanego lub uszkodzonego systemu w sieci. Jak wcześniej, warto przeczytać stronę podręcznikową (*http://www.stopspam.org/usenet/mmf/man/traceroute.html*).

Wybór sprzętu dla serwera

Jeśli pojawia się potrzeba zakupu sprzętu do obsługi witryny WWW, trzeba rozważyć kilka kwestii, zwłaszcza, czy w ogóle kupować sprzęt. O tym mowa w punkcie „Niech ktoś zrobi to za nas" na końcu tego rozdziału.

Obsługiwane platformy

Apache działa na wielu różnych platformach. Najczęściej jest uruchamiany w systemach UNIX, spośród których największą popularnością cieszą się Solaris i nieodpłatne systemy operacyjne wzorowane na Uniksie — Linux oraz FreeBSD. Warto wspomnieć także o MacOS X, choćby dlatego, że każdy komputer dostarczony z tym systemem zawiera instalację Apache.

Apache współdziała także z systemem Windows NT, choć Apache 1.3 nie działa w nim tak dobrze, jak implementacja tego serwera w systemie UNIX. Apache 2.0 jest lepiej przystosowany do platformy Windows — wykazuje większą wydajność i odporność. Trwają próby opracowania wersji Apache dla innych platform, dotychczas nie obsługujących Apache. Jednak wybór takiej platformy jest sensowny tylko, gdy mamy już sprzęt.

Korporacje mające kontrakty serwisowe i zainteresowane wsparciem technicznym powinny wybrać posiadaną platformę, zakładając, że Apache na niej działa, a wydajność i stabilność nie budzi zastrzeżeń. Dla reszty użytkowników, których budżety są ograniczone, tani komputer PC z systemem Linux lub FreeBSD jest dobrym, ekonomicznym rozwiązaniem. Obie te

platformy mają dobrą opinię pod względem stabilności. Możliwość zbudowania niedrogiego klastra tanich serwerów jest, wbrew pozorom, całkiem realna. Do grupowania komputerów w klaster może wystarczyć serwer Apache i serwer nazw.

Dla prostych serwerów z niezbyt wymagającymi witrynami WWW może wystarczyć nawet stary komputer 486 Jeśli mamy stary sprzęt i chcemy odłożyć decyzję o zakupie do czasu aż będziemy wiedzieć, jaki sprzęt jest potrzebny, stare wycofane maszyny mogą nadawać się doskonale. Warto najwyżej kupić tani komputer PC dla potrzeb programowania. okres przejściowy. Możemy skorzystać ze wzrostu mocy i spadku cen, jeśli wstrzymamy się z poważnymi zakupami jak najdłużej.

W krainie darmowego i łatwo dostępnego oprogramowania Linux i FreeBSD cieszą się jednakową popularnością. FreeBSD jest nieco bardziej stabilny i zapewnia szybszą obsługę sieci, natomiast dla systemu Linux opracowano znacznie więcej oprogramowania. Rozróżnienie jest jednak dość chwiejne, ponieważ Linux okazuje się stabilny dla większości zastosowań WWW, a przeniesienie oprogramowania do FreeBSD jest zwykle bardzo proste.

Jeśli uznamy, że stabilność jest dla nas najważniejsza i nie zamierzamy instalować zbyt dużo dodatkowego oprogramowania, nasz wybór powinien paść na FreeBSD. Jeśli planujemy zainstalowanie dodatkowych pakietów do obsługi baz danych, zabezpieczeń lub handlu elektronicznego, odpowiednim systemem wydaje się Linux. Używane dla serwerów WWW odmiany BSD to OpenBSD, NetBSD i oczywiście MacOS X.

W czasie, gdy powstawała ta książka, serwer Apache był obsługiwany na następujących platformach:

AIX	A/UX	BS2000/OSD
BSDI	DGUX	DigitalUNIX
FreeBSD	HP-UX	IRIX
Linux	MacOS X	NetBSD
Netware	OpenBSD	OS/2
OSF/1	QNX	ReliantUNIX
SINIX	Solaris	SunOS
UNIXWare	Windows 9x i ME	Windows NT, 2000 i XP

Podstawowe wymagania serwera

Jeśli pracujemy w jednolitym środowisku firmy lub instytucji ma sens użycie tego samego rodzaju sprzętu, co już istniejący, chociażby dla zachowania równowagi psychicznej administratorów i uproszczenia administracji siecią.

Jednak nie jest to argument aż tak ważny, jak mogłoby się wydawać. Jeśli serwer nie zależy silnie od sieci intranet firmy (nie wymaga, na przykład, dostępu do bazy danych), niezłym rozwiązaniem ze względu na bezpieczeństwo jest całkowite odizolowanie go od tej sieci. Ponieważ nie ma komunikacji pomiędzy serwerem WWW a innymi serwerami, kwestie zgodności nie są istotne.

Serwer Apache można uruchomić niemal na każdym sprzęcie, zatem jeśli nie ma jakichś specjalnych powodów by wybrać określonego dostawcę sprzętu, należy zdecydować się na tani lub niezbyt drogi komputer PC, który spełni swoje zadanie, o ile będzie niezawodny. Stabilności jest znacznie ważniejsza od marki.

Uruchomienie serwera na wyspecjalizowanym sprzęcie

Warto wszakże pamiętać o jednym. Serwer Apache najlepiej eksploatować na sprzęcie przeznaczonym tylko do tego celu. Mając na uwadze wymagania serwera WWW wobec procesora, dysku oraz sieci i wiedząc, że do uruchomienia Apache wystarcza tani sprzęt, nie ma powodu, by nie kupić osobnego serwera do obsługi witryn WWW i uniknąć dzielenia zasobów z innymi aplikacjami. Nie jest też dobrym pomysłem przeznaczanie do szerokiego, publicznego dostępu WWW komputera, który zawiera ważne aplikacje i pliki.

Serwery o wysokiej wydajności lub wysokiej niezawodności

Dla wymagających aplikacji warto wziąć pod uwagę komputer wieloprocesorowy. Mając rozszerzalny system, można zachować wydajność serwera poprzez dołożenie dodatkowych procesorów lub pamięci, gdy wymagania wzrosną.

Inne rozwiązanie godne zastanowienia, zarówno z punktu widzenia kosztów, jak i niezawodności, to klaster kilku niezależnych komputerów, które mogą posłużyć do utworzenia wirtualnego serwera. Kilka znanych rozwiązań, jak i nietypowe klastry, omówiono w rozdziale 7.

Pamięć operacyjna

Nigdy dość pamięci. Im więcej pamięci mamy do dyspozycji, tym więcej danych można w niej przechować, co umożliwia szybki dostęp. Dotyczy to nie tyko Apache, ale i każdego innego procesu, który uruchamiamy na komputerze — na przykład aplikacji dla bazy danych.

Dokładniej rzecz ujmując, ilość niezbędnej pamięci to ilość pamięci, która umożliwia serwerowi i innym usługowym procesom działanie bez uciekania się do pamięci wirtualnej. Jeśli system operacyjny stwierdzi niedostatek pamięci, będzie musiał tymczasowo przesunąć dane z pamięci operacyjnej na dysk (proces zwany **wymianą**. Kiedy dane te będą znowu potrzebne, zostaną wymienione z czymś innym w pamięci — chyba że od ostatniej wymiany zwolniła się brakującą ilość pamięci.

Jasne, że takie działanie nie jest zbyt wydajne. Procesy zależne od wymienianych danych zostają wstrzymane, dysk i procesor są zajęte przekazywaniem danych z pamięci na dysk i z powrotem. Traci na tym obsługa stron WWW. Niekorzystny wpływ na wydajność może mieć zwłaszcza wymiana bufora serwera WWW lub często czytanych tabel bazy danych.

Aby oszacować ilość potrzebnej pamięci, należy dodać ilość pamięci zajmowaną przez każdą z aplikacji i wziąć tę sumę. Jest to niezbyt dokładna metoda, więc w mocy pozostaje reguła „dodaj więcej pamięci". Jedynie badając działanie serwera można określić, czy ilość dostępnej pamięci jest wystarczająca.

Narzędzie `vmstat` w większości odmian systemów UNIX jest jednym z tych programów, które umożliwiają monitorowanie deficytu pamięci w serwerze oraz czasu spędzonego na wymianie danych pomiędzy pamięcią a dyskiem. Podobne narzędzia są dostępne także na innych platformach. Windows NT np. zawiera bardzo dobre narzędzie o nazwie `perfmon`.

System operacyjny sprawnie zarządzający pamięcią jest równie ważny (patrz punkt „Zestawienie właściwości systemu operacyjnego" w dalszej części tego rozdziału).

Interfejs sieciowy

Wydajność procesora i wielkość pamięci same w sobie są niewystarczające. Często wąskim gardłem jest niedostateczna wydajność wejścia-wyjścia systemu (częstotliwość dostępu do karty sieciowej i dysku twardego).

W sieci intranet spore wymagania ma z jednej strony sieć, a z drugiej karta sieciowa. Zwykłe łącza 10Base2 lub 10BaseT mogą sprawiać problemy. Sieć *10Base* ma przepustowość od 6 do 8 megabitów na sekundę, ale serwer WWW, do którego wykonuje się 90 połączeń na sekundę, z łatwością osiąga tę wartość.

Ponieważ karty sieciowe i kable 100baseT ostatnio potaniały, nie ma powodu, by inwestować w sieć *10Base*, chyba, że chcemy zadbać o zgodność ze starszym sprzętem. Nawet w takim przypadku dobrze będzie zastosować karty sieciowe *10/100Base* — zawsze można unowocześnić resztę sieci w późniejszym terminie. Dla najbardziej wymagających aplikacji mamy do dyspozycji standard Gigabyte Ethernet, koszt jego wprowadzenia jest jednak niebagatelny.

Zakładając, że inne komputery nie obciążają niepotrzebnie sieci, użycie zwykłej karty Ethernet będzie w większości przypadków wystarczające. Jedyny warunek jest taki, by nie wykorzystywać do tego celu najtańszej karty ze sklepu komputerowego ze sprzętem po okazyjnej cenie. Trzeba pamiętać, że karty z „niższej półki" nie wykorzystują takich własności jak bezpośredni dostęp do pamięci (DMA; inne oznaczenie to *bus-mastering*) i mają gorszą wydajność niż droższe karty, mimo że są z nimi „kompatybilne".

Podwójne interfejsy sieciowe

Doskonałym rozwiązaniem dla serwerów, zwłaszcza tych, które zamierzamy podłączyć zarówno do serwera dostawcy usług internetowych, jak i do lokalnej sieci wewnętrznej, są dwie karty sieciowe z różnymi adresami IP w różnych sieciach. Interfejs sieci zewnętrznej przeznaczony jest wyłącznie dla dostępu do serwera WWW, a interfejs sieci wewnętrznej służy do połączeń z bazą danych lub systemem tworzenia kopii zapasowych. Dzięki temu można przetwarzać żądania do bazy danych lub sporządzać kopie zapasowe bez wpływu na przepustowość zewnętrznego łącza, a popularna witryna WWW nie wywiera ujemnego wpływu na sieć wewnętrzną.

Podwójne interfejsy poprawiają zabezpieczenie systemu — izolacja sieci wewnętrznej od zewnętrznej i wykluczenie wyboru tras pomiędzy nimi pozwala łatwo ograniczyć dostęp zewnętrznych użytkowników do sieci wewnętrznej. Przykładowo: można to zrealizować instalując zaporę (*firewall*) pomiędzy interfejsem dla sieci wewnętrznej a resztą sieci, pozostawiając na zewnątrz jedynie serwer WWW.

Połączenie internetowe

Jeśli serwer ma być podłączony do Internetu, należy szczegółowo rozważyć zarówno typ połączenia, które mamy użytkować, jak i możliwości dostawcy usług internetowych

Oto kilka pytań, na które należy odpowiedzieć, rozważając wybór dostawcy usług internetowych:

- Czy dostawca usług internetowych jest solidny?
- Czy ma dobrą łączność z Internetem (kto jest jego partnerem, czy dysponuje nadmiarowymi łączami)?
- Czy będziemy dzielić pasmo z innymi użytkownikami?
- Jeśli tak, to czy dostawca usług internetowych może nam zaoferować wydzielone łącze?

Jeśli zamierzamy uruchomić witrynę o powiązaniach międzynarodowych (na przykład, gdy zamierzamy prowadzić wszystkie regionalne witryny międzynarodowej firmy z jednego miejsca), dodatkowym pytaniem będzie:

- Czy usługodawca ma globalny dostęp?

Na zakończenie, można spróbować zadać pytanie o kompetencje dostawcy:

- Czy serwis techniczny zna odpowiedź na wszystkie te pytania?

Należy pamiętać, że sama oferta szerokiego pasma nie oznacza, że użytkownicy Internetu będą mogli ją wykorzystać. Zależy to w dużym stopniu od tego, jak dobrze usługodawca jest zintegrowany ze swoimi partnerami i siecią szkieletową. Wielu dostawców usług internetowych ma dostęp u jednego dostawcy i jeżeli ten dostawca jest przeciążony, szerokie pasmo nie przyda się ani nam, ani odwiedzającym nasz serwer klientom, nawet gdy szerokość pasma wychodzącego jest teoretycznie bardziej niż zadawalająca.

Dysk twardy i kontroler

Szybkie dyski twarde i dostosowane do nich kontrolery powinny zdecydowanie znaleźć się w komputerze przeznaczonym dla serwera WWW. Jeśli w dodatku wydajność jest dla nas sprawą kluczową, zdecydowanie lepiej będzie skorzystać z kontrolera SCSI, a nie IDE.

W przypadku witryn WWW, które będą odwiedzane często, znacznie rozsądniejsze jest użycie kilku mniejszych dysków niż jednego dużego. Jeśli, na przykład, jest obsługiwana jedna duża baza danych lub kilka serwerów wirtualnych, zaleca się, by dane były przechowywane na osobnych dyskach. Pozwoli to na lepszą wydajność w obsłudze witryny WWW, bo dysk może w danej chwili czytać dane tylko w jednym miejscu na raz.

Dla zwiększenia wydajności dysków można użyć macierzy RAID 0 (*striping*). Po dodaniu RAID 1, co zapewnia redundancję, może to być wysoce efektywny sposób zwiększenia wydajności serwera. Taki system określa się rozmaicie: jako RAID 0+1, RAID 1+0 i RAID 10 — wszystkie te nazwy oznaczają to samo. Tego typu rozwiązanie może być jednakże dość kosztowne.

Zestawienie właściwości systemu operacyjnego

Aby serwer działał efektywnie (to znaczy aby działał zarówno stabilnie, jak i sprawnie), system operacyjny hosta musi nadawać się do tego zadania. Omówiliśmy przydatność systemów operacyjnych dla serwera Apache i wspomnieliśmy, że do instalacji Apache na ogół zaleca się UNIX. Jednak bez względu na to, który system operacyjny zostanie wybrany, powinien on spełniać w jakimś stopniu następujące kryteria:

- **Stabilność**

  System operacyjny powinien być niezawodny i zdolny do dowolnie długiej pracy bez ponownego rozruchu. Złe zarządzanie pamięcią jest zasadniczym powodem niestabilności systemu, która objawia się po dłuższym czasie użytkowania.

- **Bezpieczeństwo**

  System operacyjny powinien być odporny na wszystkie rodzaje ataków, włącznie z atakami DOS — w wyniku takiego ataku system jest zajęty, a to powoduje odmowę usługi dla uprawnionych użytkowników. System operacyjny powinien mieć także udokumentowaną historię w zakresie bezpieczeństwa. Ujawnione luki w zabezpieczeniach powinny być szybko usuwane przez odpowiedzialne za to organy. W tym przypadku szybka reakcja na ujawnione zagrożenie oznacza dni, a nie tygodnie.

- **Wydajność**

  System operacyjny powinien efektywnie korzystać z zasobów, obsługując pracę w sieci bez zbędnego obciążenia dla reszty systemu operacyjnego. Powinien sprawnie przełączać pomiędzy zadaniami. Apache w tworzy wiele procesów do obsługi przychodzących połączeń. Niesprawne przełączanie kontekstu powoduje spadek wydajności serwera WWW. Jeśli planujemy uruchomić serwer na wieloprocesorowym komputerze, warto rozważyć zagadnienia wydajności SMP.

- **Utrzymanie**

  System powinien być łatwy do uaktualnienia lub nałożenia „łaty" usuwającej luki w zabezpieczeniach. Nie powinien wymagać przy tym ponownego rozruchu ani przejścia w tryb *offline*, chyba że to jest jakaś zasadnicza instalacja lub uaktualnienie. Nie powinien wymagać ponownego rozruchu dla utrzymania lub uaktualnienia jedynie pewnego swego elementu.

- **Pamięć operacyjna**

  System operacyjny powinien efektywnie wykorzystywać pamięć, unikać wymiany danych pomiędzy pamięcią a plikiem na dysku, chyba że jest to absolutnie konieczne — wtedy jednak taka operacja powinna minimalnie obciążać system. System operacyjny nie powinien ulegać „wyciekom pamięci". Oprogramowanie powodujące wycieki pamięci jest jednym z zasadniczych powodów, dla których serwery WWW mogą być zawodne. Na przykład, do niedawna system Windows NT nie miał najlepszej opinii pod tym względem — takie problemy stwarzały również aplikacje. Na szczęście Apache do nich nie należy, choć starsze wersje nie były bez wad.

- Poważniejsze problemy mogą stwarzać niezależne moduły, jednak serwer Apache umożliwia wymuszenie okresowego wznawiania procesów dzięki dyrektywie `MaxRequestsPerChild`. co ogranicza szkody, jakie moduły te mogą powodować w systemie. Jeżeli planujemy duże aplikacje, takie jak serwery baz danych, lepiej sprawdzić jaką mają opinię.

Nadmiarowość i archiwizacja danych

Jeśli zamierzamy uruchomić serwer o jakimkolwiek znaczeniu, powinniśmy poświęcić nieco uwagi temu, w jaki sposób przywrócić nasz serwer do działania w przypadku, gdy z jakiegoś powodu ulegnie awarii. Na przykład może wystąpić awaria sprzętu komputerowego lub w wyniku włamania do systemu dane zostaną utracone lub ich poufność naruszona. Dobrą „pierwszą linią obrony" jest macierz dysków RAID, ale wadą tego rozwiązania jest jego względnie duży koszt. Macierz dysków pozwala przechować kopię zapasową oprogramowania samego serwera komputerowego — jest to jednak niewielka pociecha w razie pożaru lub eksplozji komputera (jest to mało prawdopodobne, ale jednak możliwe).

Aby zapewnić *archiwizację* wystarczy wyposażyć komputer w napęd taśmy cyfrowej DAT lub inne urządzenie pamięci masowej i odpowiednio skonfigurować serwer tak, by kopiował właściwe pliki na taśmę lub inny nośnik w regularnych odstępach czasu. Taką archiwizację można wykonać nawet bez specjalistycznego oprogramowania do archiwizacji i tworzenia kopii zapasowych — prosta usługa `cron` zapewnia okresowe realizowanie zaplanowanych zadań.

Znacznie lepszym rozwiązaniem jest archiwizacja za pośrednictwem lokalnej sieci. Pozwala to na kopiowanie danych na zapasowy komputer, który jest gotów rozpocząć pracę w przypadku, gdy serwer podstawowy przestanie działać. Wyklucza to również konieczność interwencji operatora przy wymianie taśm DAT (lub innych nośników) w napędzie urządzenia do archiwizacji.

Jeśli serwer jest dostępny za pośrednictwem Internetu (a nawet wtedy, gdy nie jest), powinniśmy również podjąć środki zapobiegające próbom włamania do systemu. Jeśli takie włamanie już się zdarzyło, powinniśmy przywrócić wszystko z zaufanych kopii zapasowych i archiwów. Procedura ta obejmuje ponowną instalację systemu operacyjnego, konfigurację systemu i przywrócenie witryn WWW z kopii zapasowych. Jeśli tworzymy kolejne kopie zapasowe na jednym i tym samym nośniku codziennie, może się zdarzyć, że zanim spostrzeżemy problem, zapiszemy ponownie kopię zapasową, usuwając jej poprzedni zapis — okaże się wówczas, że nie mamy dobrej kopii zapasowej, sporządzonej przed pojawieniem się problemów. Morał jest taki, że należy przechowywać nie tylko najnowszą kopię zapasową, ale i starsze, starannie datowane kopie.

Jest kilka komercyjnych narzędzi do robienia kopii zapasowych i archiwizowania danych. Wybór może być podyktowany środowiskiem, w jakim pracuje serwer lub strategią archiwizowania przyjętą w danej firmie. Darmowe alternatywy obejmują oczywiste, choć mało finezyjne narzędzia, takie jak FTP lub NFS do kopiowania drzewa katalogowego z jednego serwera na drugi (jeśli nie ma takiej konieczności, powinniśmy raczej unikać uaktywniania na serwerze ryzykownego z punktu widzenia bezpieczeństwa sieciowego systemu plików NFS).

Lepszym, nieodpłatnym narzędziem do robienia zapasowych kopii jest `rsync` — „inteligentna" wersja standardowego w systemie UNIX polecenia `rcp`. Narzędzie to pozwala na kopiowanie samych różnic w hierarchii katalogów. Co więcej, potrafi działać z wykorzystaniem szyfrowanego połączenia przez `ssh` (*secure shell*) — kolejne nieodpłatne narzędzie. W przypadku tworzenia kopii zapasowych plików serwera za pośrednictwem Internetu, należy poważnie rozważyć zastosowania tego rozwiązania. Zarówno `rsync`, jak i `ssl` są omówione w rozdziale 10. Warto wspomnieć przy okazji o systemie wersji CVS (*Concurrent Versioning System*). Częściej jest on stosowany dla kodu źródłowego, ale dobrze nadaje się dla plików HTML. Więcej informacji na temat CVS można znaleźć pod adresem *http://www.cvshome.org/*.

Jeszcze jedna uwaga na zakończenie tematu archiwizacji za pośrednictwem sieci. Nawet jeśli stosujemy „inteligentny" system, który jest zdolny do przyrostowego robienia kopii zapasowych, musimy pamiętać, że obszerna witryna WWW wymaga przesyłania sporych ilości danych. Jeśli do tego serwer jest obciążony, dane te zapełnią podczas robienia kopii dostępne pasmo, które w innym przypadku mogłyby zostać wykorzystane do obsługi żądań przeglądarek. Jest więc istotne, by dobrze zaplanować archiwizowanie danych i wykonywać je w odpowiednim czasie. Jeśli mamy dużo danych do przekopiowania, lepiej kopiować je etapami (na przykład katalog po katalogu), uwzględniając tylko te elementy systemu plików, które uległy zmianie. Podwójne połączenie sieciowe jest w takim przypadku dużym udogodnieniem, gdyż pozwala na archiwizowanie danych za pośrednictwem wewnętrznego interfejsu sieciowego, podczas gdy zewnętrzna sieć pozostaje w gotowości do obsługi nadchodzących żądań HTTP.

Specyficzne rozwiązania sprzętowe

Kilku dostawców sprzedaje sprzęt komputerowy z fabrycznie zainstalowanym serwerem Apache lub jakimś innym oprogramowaniem opartym na tym serwerze. Na razie wszystkie te serwery są oparte na systemach UNIX, a zwłaszcza Linux. Kilku dostawców usług internetowych znanych jest również z tego, że oferuje do kupienia lub wynajęcia serwery, które są specjalnie przeznaczone dla potrzeb Apache.

Komputery Cobalt Qube i RaQ firmy Sun

Cobalt była jedną z pierwszych firm projektujących sprzęt komputerowy przeznaczony dla systemu Linux. Specjalna seria takich mikroserwerów, wyróżniająca się obudowami o charakterystycznym, kobaltowym odcieniu koloru niebieskiego, stała się popularna wśród firm komercyjnych i dostawców usług internetowych.

Obecnie wchłonięta przez Sun Microsystems, firma Cobalt oferuje zasadniczo dwa produkty. Pierwszy z nich, przeznaczony dla odbiorców biznesowych, to system *Qube 3*, wymagający ponoszenia niewielkich kosztów związanych ze sprzętem — obudowa tego komputera, zgodnie z tym, co sugeruje jego nazwa, ma kształt kostki. Drugi system to *RaQ* — serwer do instalacji na stojakach. Opracowywano go przede wszystkim z myślą o dostawcach usług internetowych, którzy chcą montować całe tuziny serwerów na jednym stojaku. Te dwa systemy są do siebie bardzo podobne pod względem oprogramowania.

Starsze wersje tych systemów były oparte na procesorze MIPS, nowsze korzystają z procesorów Intela. Zarówno Sun Cobalt Qube, jak i Sun Cobalt RaQ, mają zainstalowaną okrojoną wersję systemu operacyjnego Red Hat oraz narzędzia konfiguracji oparte na serwerze WWW, które można wykorzystać do zdalnej konfiguracji serwera. Jest to ogromna zaleta z punktu widzenia administratora systemu. Konfiguracja witryn WWW, hostów wirtualnych, poczty elektronicznej i systemu nazw DNS może być przeprowadzona za pomocą przeglądarki. Na serwerach Cobalt są uruchomione dwie kopie serwera Apache jednocześnie — jedna do obsługi witryny WWW, a druga do jej konfiguracji. To sprytne rozwiązanie umożliwia działanie serwera Apache przeznaczonego do administrowania, nawet wtedy, gdy zasadnicza konfiguracja ulegnie zniszczeniu.

Starsze serwery z procesorem MIPS, których używane egzemplarze można nabyć po okazyjnej cenie, mają pewien mankament. Otóż, można mieć trudności ze skompletowaniem potrzebnego dodatkowego oprogramowania, gdyż platforma ta nie należy do najpopularniejszych. Dla procesora MIPS jest bardzo mało oprogramowania dostępnego w postaci binarnej, znacznie mniej niż dla platform Intel. Dlatego starsze serwery z procesorami MIPS, nadają się do takich zastosowań, które nie wymagają większej ilości dodatkowego. Jeśli myślimy o instalacji dodatkowych aplikacji, będziemy musieli to robić prawdopodobnie poprzez kompilację kodu źródłowego. Trzeba przy tym pamiętać, że Red Hat ułatwia to zadanie, gdyż stosuje pakiety RPM, których szeroki wybór jest dostępny pod adresem *http://www.rpmfind.net/*.

Więcej szczegółowych informacji o komputerach Cobalt można znaleźć na stronie *http://www.cobalt.com/*.

IBM

IBM oferuje szereg komputerów, które mają fabrycznie zainstalowany własny serwer HTTPD — jest to serwer Apache z etykietą IBM i numerem wersji, która odpowiada oryginalnej wersji Apache. Serwer ten jest ściśle powiązany z serwerem aplikacji IBM WebSphere, dostępnym dla platform AIX, Solaris, Linux oraz Windows.

Technologia Java Server Pages i serwletki również zostały włączone przez IBM do pakietu HTTPD. Są to technologie oparte na innym projekcie Apache Software Foundation o nazwie Jakarta Tomcat (*http://jakarta.apache.org/*). Projekt ten stanowi alternatywę dla tworzenia treści dynamicznej, pozwalającą uniknąć pewnych trudności związanych z integracją z serwerem. Omówimy to w rozdziale 12., przy okazji prezentacji rozszerzenia Tomcat dla serwera Apache.

Więcej szczegółów na temat oferty IBM można znaleźć na stronie *http://www.ibm.com/*.

Standardowy Linux plus Apache

Linux jest obecnie fabrycznie instalowany na szeregu komputerów, które mają pełnić rolę serwerów lokalnych i serwerów do zastosowań w Internecie. Standardowy system Linux nie ma takich narzędzi konfiguracyjnych, jak RaQ, Qube lub Netwinder. Niemniej jednak, dla potrzebujących dostępna jest pomoc techniczna. Wykaz dostawców takiego sprzętu ciągle rośnie — Linux VAR HOWTO pod adresem *http://www.linuxdoc.org/HOWTO/VAR-HOWTO.html*, zawiera użyteczne wskazówki i odnośniki.

Niech ktoś zrobi to za nas

Zamiast samodzielnie konfigurować serwer, spędzając długie godziny na zmaganiach z wszelkimi możliwymi problemami niezawodności, szybkości połączeń i archiwizacji danych, możemy wybrać inną drogę i kupić na własność lub wynająć dedykowany serwer u naszego ISP, co nazywa się kolokacją.

Zalety takiej decyzji są oczywiste — usługodawca zajmuje się codziennym utrzymaniem serwera, natomiast nam pozostaje cieszyć się możliwościami, jakie oferuje serwer używany na wyłączność. Można nawet ponownie skompilować i skonfigurować serwer Apache tak, jak sobie tego życzymy, ponieważ sprawujemy całkowitą kontrolę nad komputerem. Oznacza to także, że możemy przyczynić się do jego zniszczenia. Sama fizyczna nieobecność serwera w zasięgu naszego wzroku nie oznacza, że można zaniedbywać obowiązki administratora serwera WWW.

Wadą takiego rozwiązania jest fizyczne oddalenie od serwera. Jeżeli problem jest poważny, możemy nie mieć dostępu do serwera, żeby zobaczyć o co chodzi. Także ISP prawdopodobnie wprowadzi ograniczenia pasma, z których musimy zdawać sobie sprawę. Musimy również pamiętać, że polegamy na jakości usług naszego ISP — warto więc najpierw sprawdzić, jak sprawna jest oferowana przez niego pomoc techniczna.

Ponadto, faktyczny zakres usług realizowanych przez rozmaitych dostawców internetowych może być odmienny. Przykładowo, niektórzy archiwizują automatycznie pliki serwera, a inni nie. Tak jak ze wszystkimi rzeczami dostępnymi przez Internet, dobrze jest zatem sprawdzić przyszłego dostawcę szukając wzmianki o nim na listach i grupach dyskusyjnych Usenet. Lepiej nie kupować kota w worku!

> Więcej materiałów, przeznaczonych szczególnie dla początkujących, można znaleźć pod adresem: *http://httpd.apache.org/docs/misc/FAQ.html#what*.