

Основы разработки Web-приложений

Бекасов Денис Евгеньевич
bekasov@bmstu.ru

План

```
function plan() {  
    'Веб и Интернет';  
    'Протоколы';  
    'Сервер и клиент';  
    'Архитектура';  
    'Серверная разработка';  
    'Клиентская разработка';  
};
```



Web-технологии





Как работает Web?



А почему?



Классический web-разработчик



Web-технологии
Так почему же?

Так сложилось.

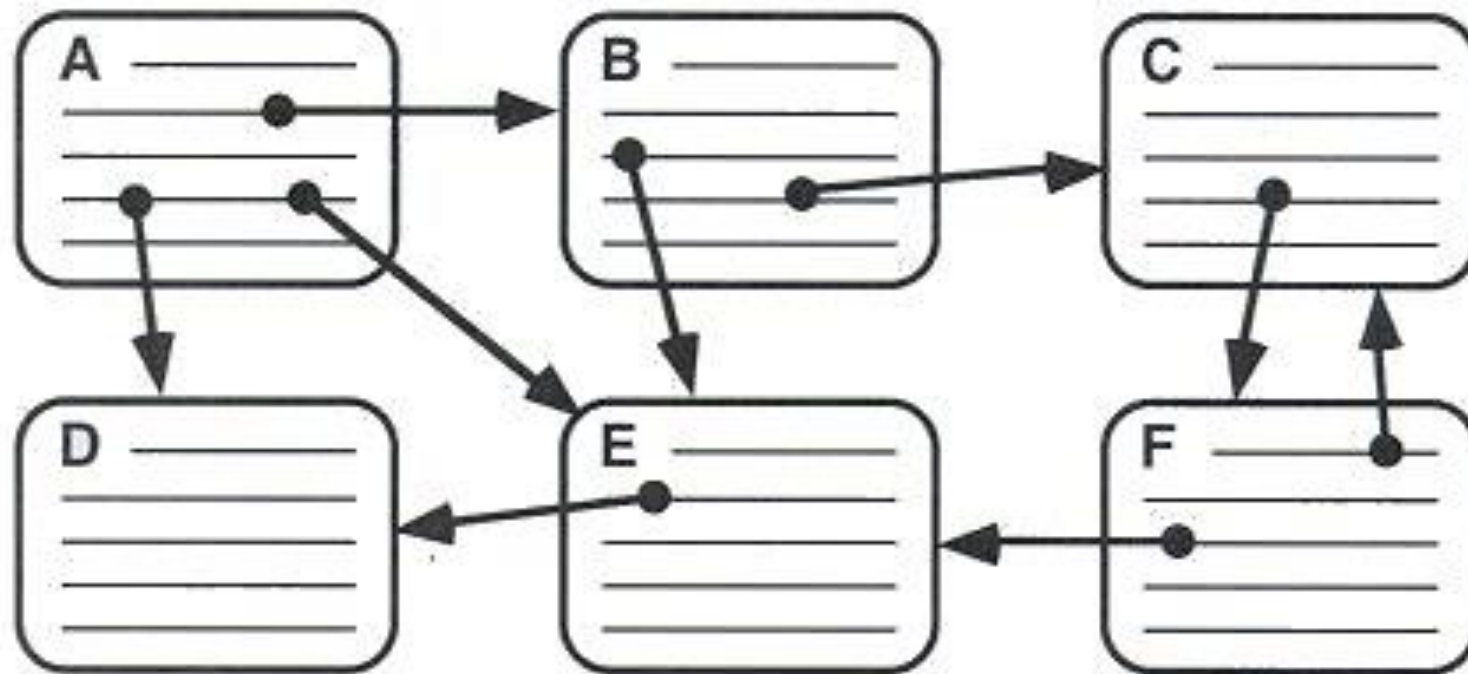
Веб-приложение



Веб (World Wide Web)



Гипертекст



Сеть интернет



Интернет: Начало

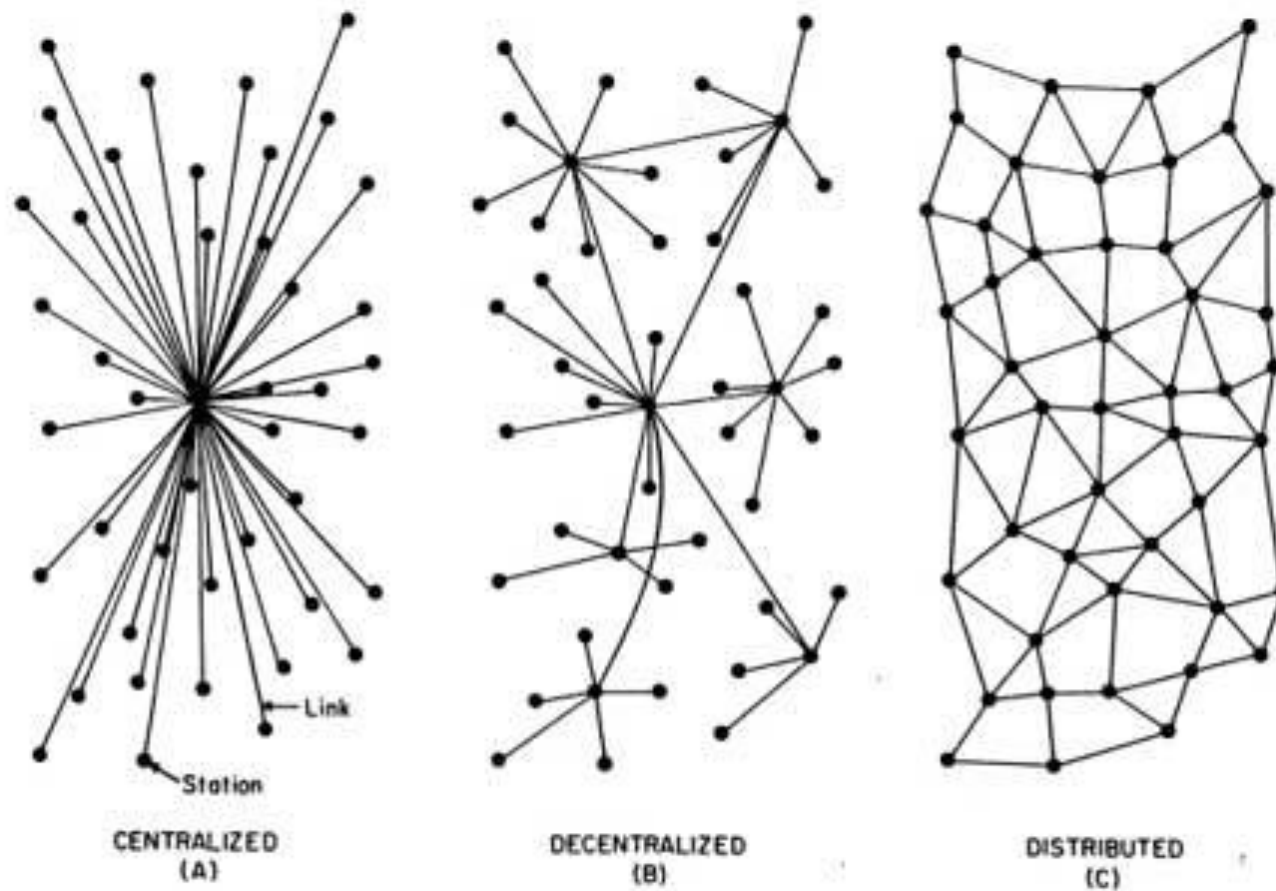
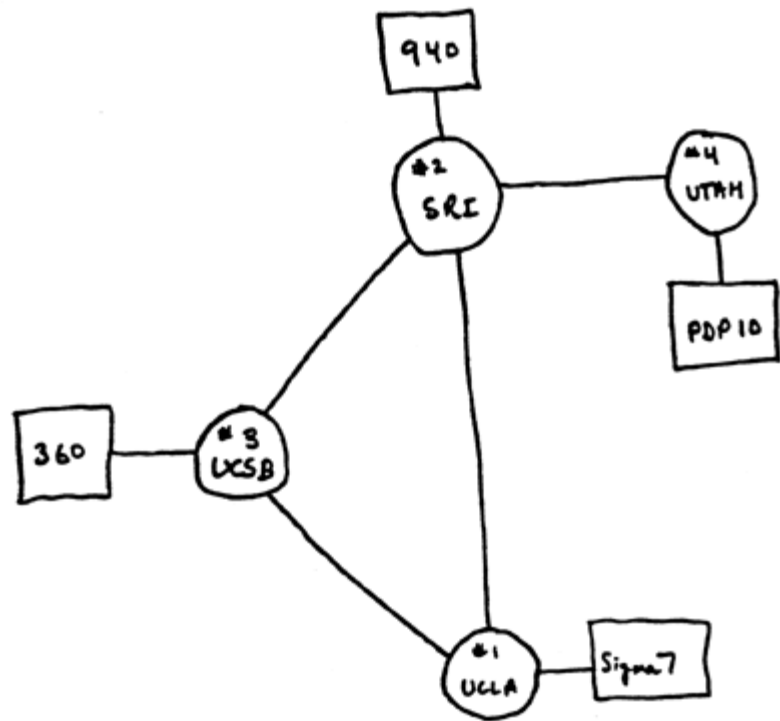


FIG. 1 – Centralized, Decentralized and Distributed Networks

ARPANET

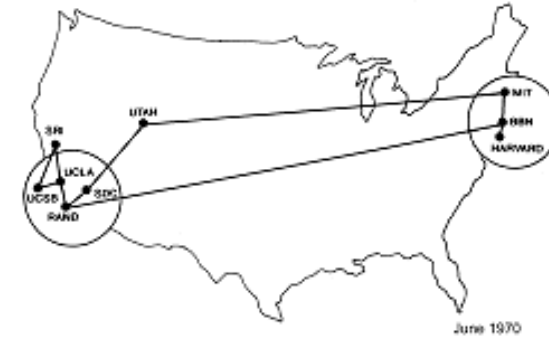
- Открытость
- Децентрализованность
- Пакетная декомпозиция

Интернет: Запуск ARPANET

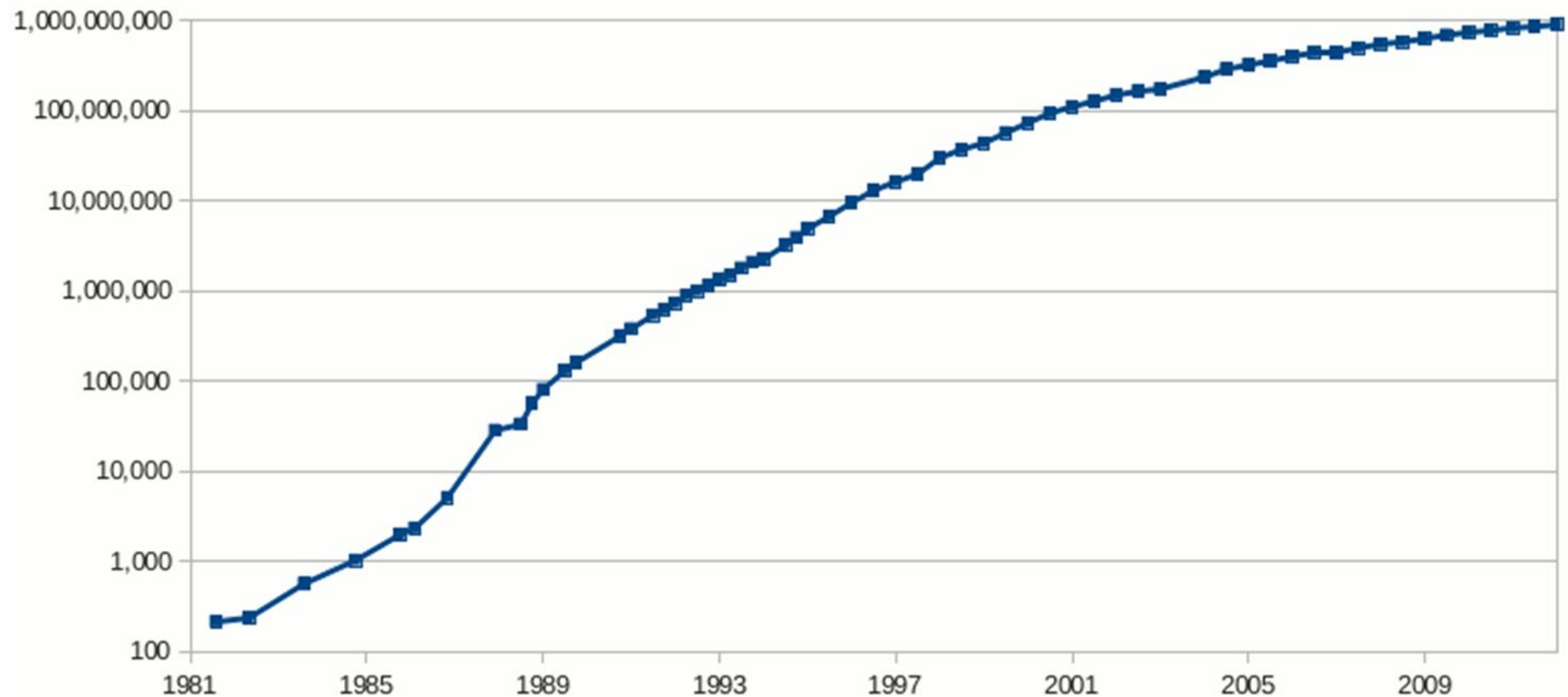


29 OCT 69	2100	LOADED OP. PROGRAM	CSK
		FOR BEN BARKER	
		BBV	
		<hr/>	
	22:30	Talked to SRI	CSK
		Host to Host	
		Left op. program	CSK
		running after sending	
		a host dead message	
		to imp.	

ARPANET



Интернет: экспонента роста хостов



Рабочая группа Интернета



Интернет

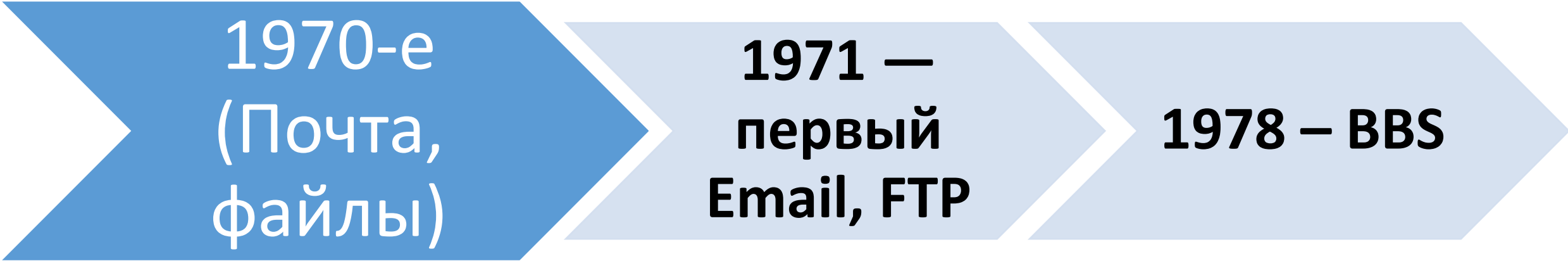


1960-е
(ARPANET)

The diagram consists of two chevron-shaped boxes pointing to the right. The first box is dark blue and contains the text '1960-е (ARPANET)'. The second box is light blue and contains the text '1969 — сеанс связи ARPANET'. The boxes are connected by a white arrow shape pointing from the first to the second.

**1969 —
сеанс связи
ARPANET**

Интернет



1970-е
(Почта,
файлы)

The diagram consists of three chevron-shaped boxes pointing to the right, connected by a continuous line. The first box is dark blue and contains the text '1970-е (Почта, файлы)'. The second and third boxes are light blue and contain the text '1971 — первый Email, FTP' and '1978 — BBS' respectively.

**1971 —
первый
Email, FTP**

1978 — BBS

Интернет

1980-е
(IP, TCP,
HTTP)

1980 – IP (RFC
760)

1981 – TCP (RFC
793), IPv4 (RFC
791)

1982 – SMTP (RFC
821)

1983 — ARPANET
переходит на
TCP/IP, Telnet
(RFC 854)

1980-е
(IP, TCP,
HTTP)

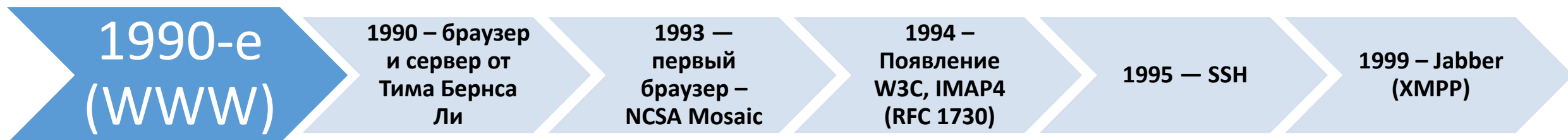
1984 – POP (RFC
918), DNS

1986 – IMAP,
NNTP

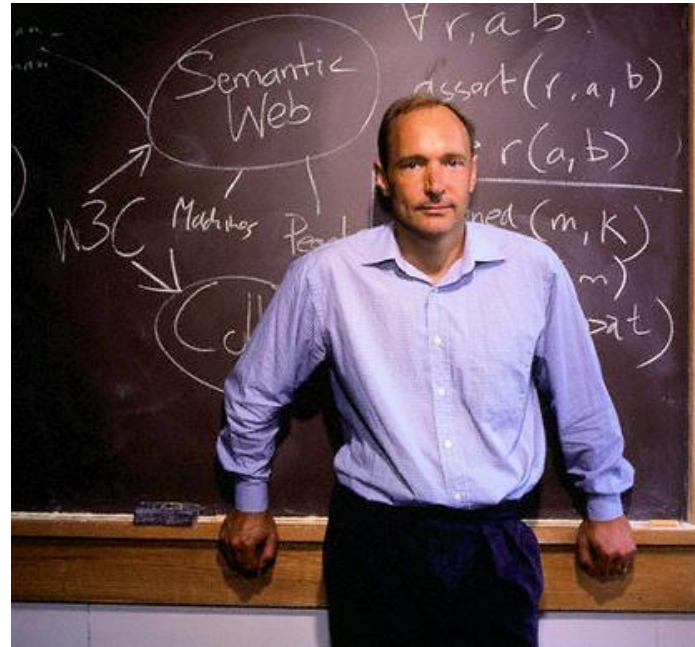
1988 – POP3 (RFC
1081)

1989 — WWW,
HTTP, HTML

Интернет



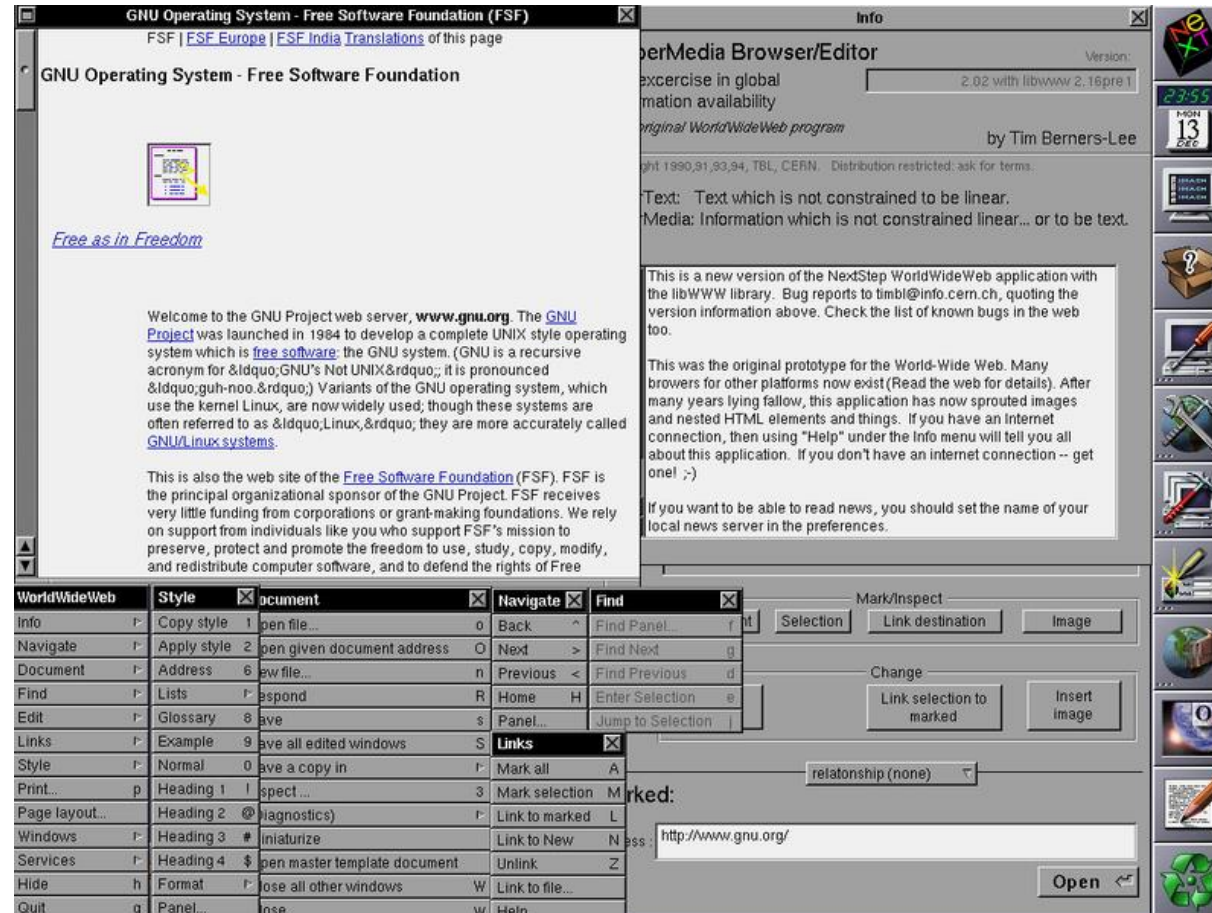
WWW, Церн и Тим Бернерс-Ли



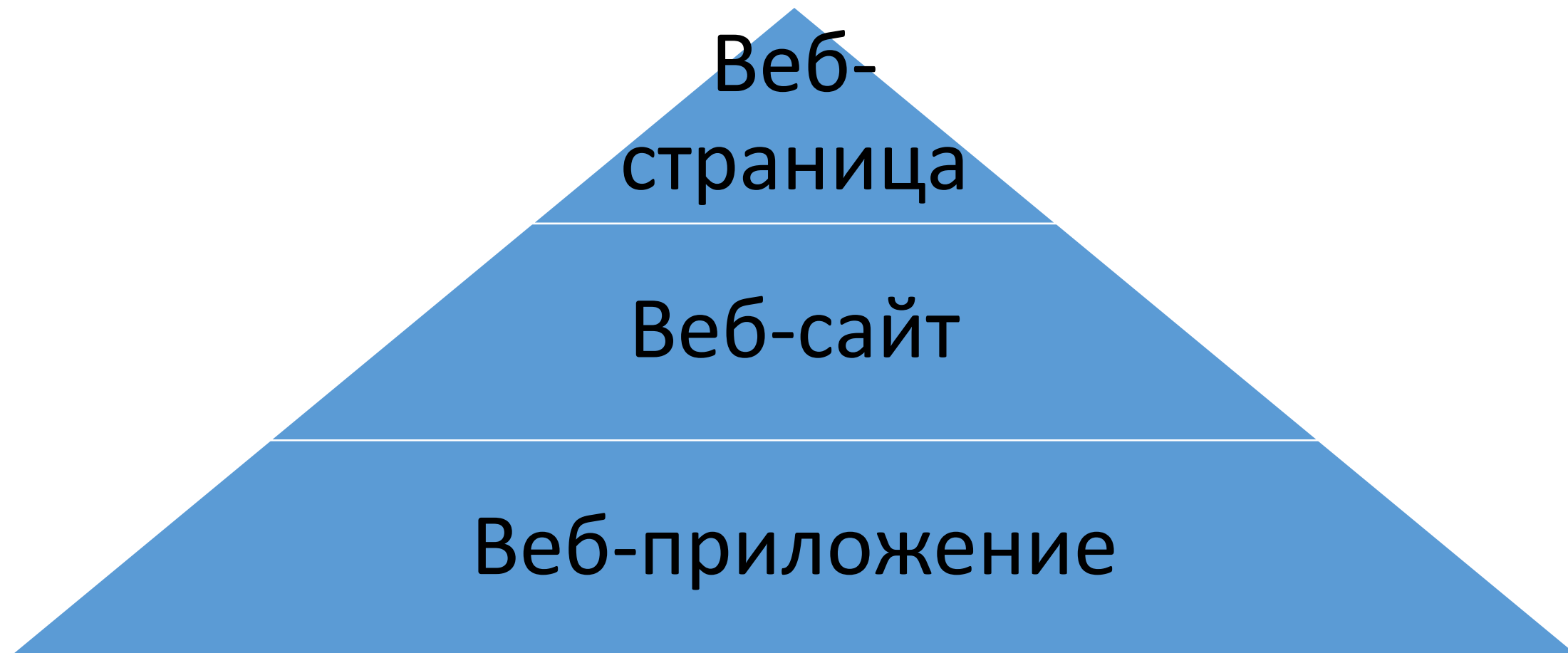
Всемирная сеть



WorldWideWeb



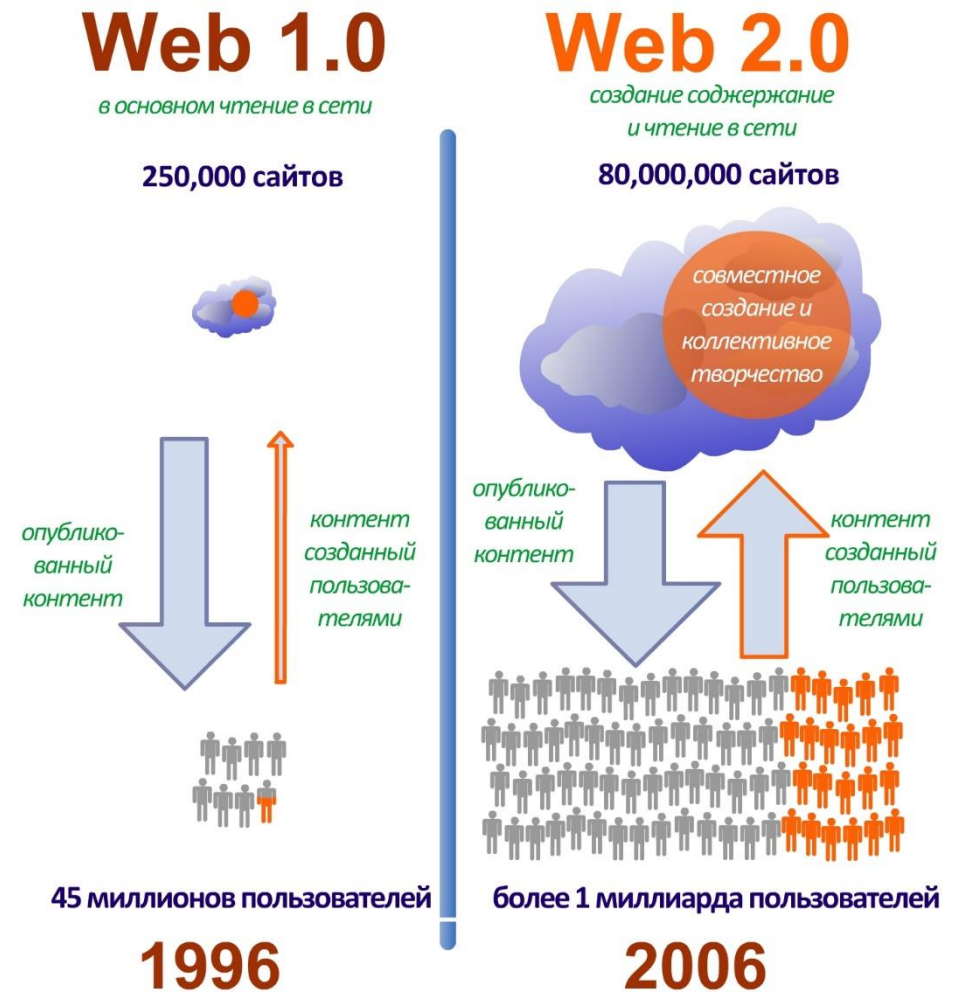
Гипертекстовые документы



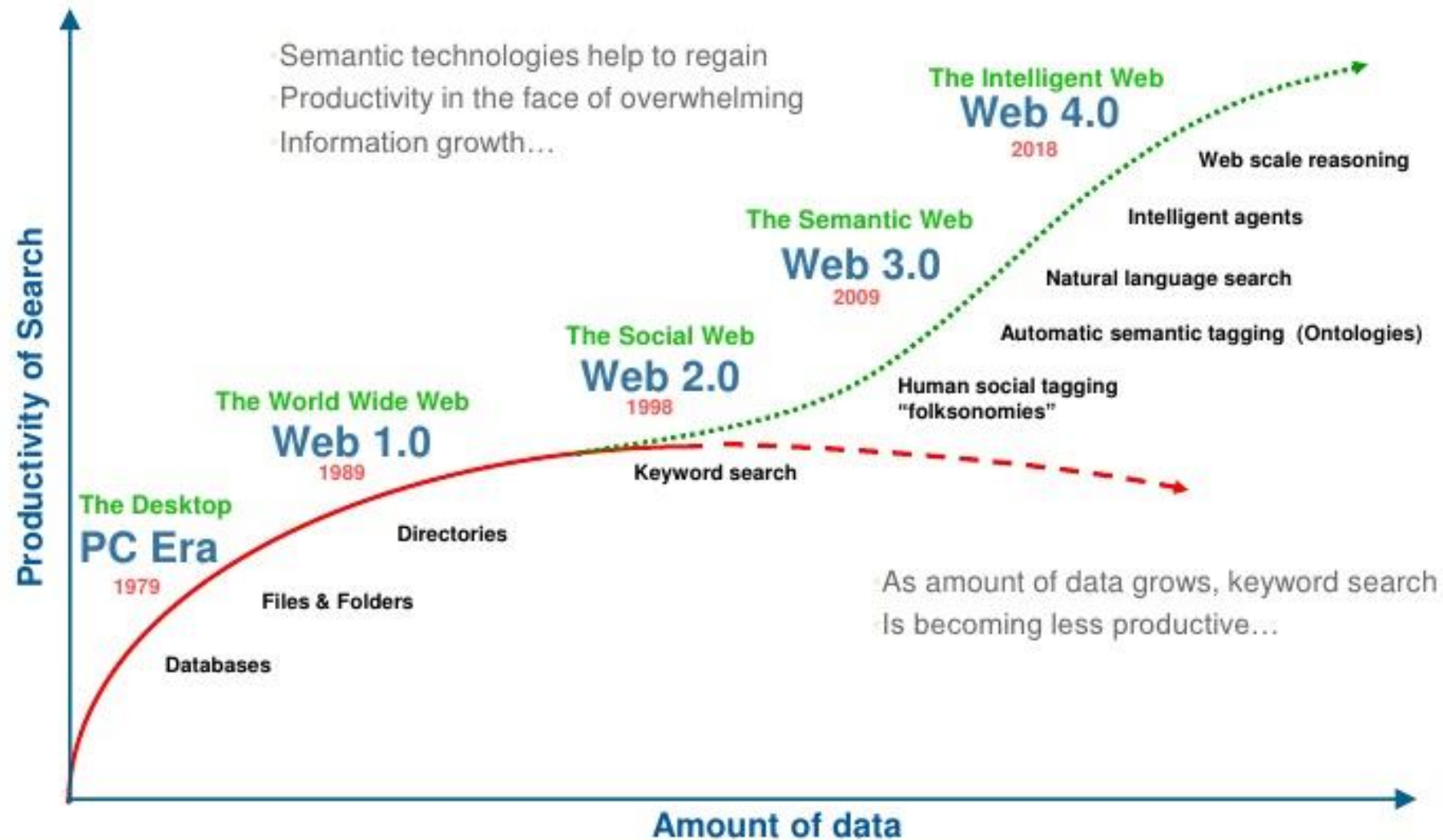
Web 1.0 vs Web 2.0



Тим О'Рейли «Что такое Web 2.0»



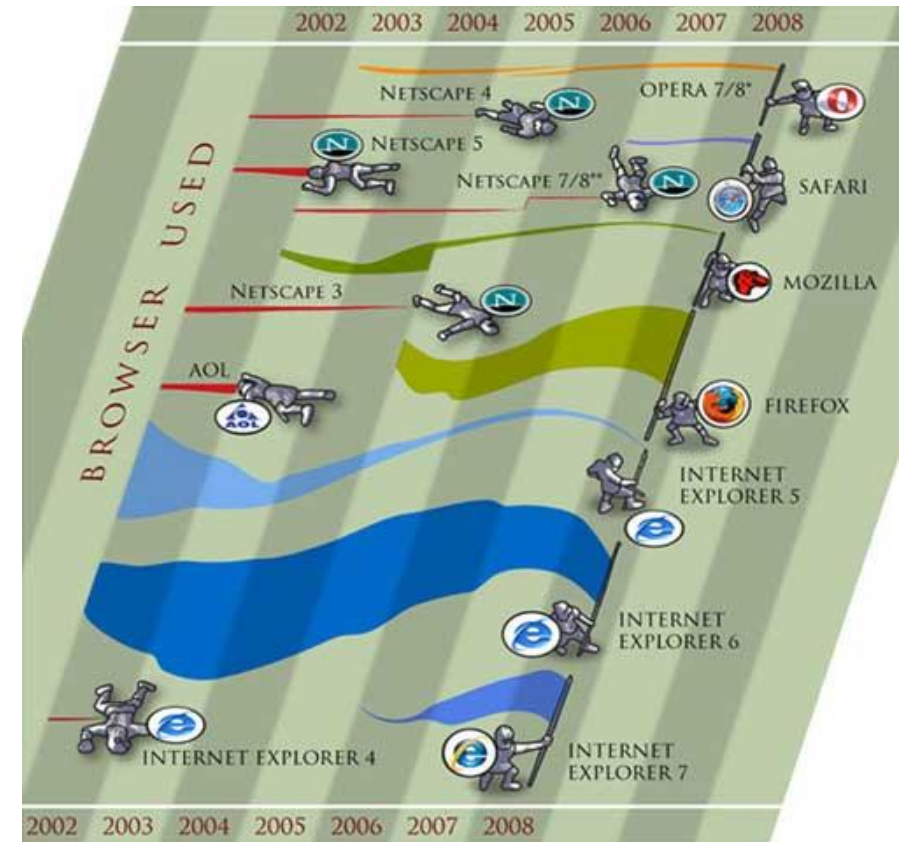
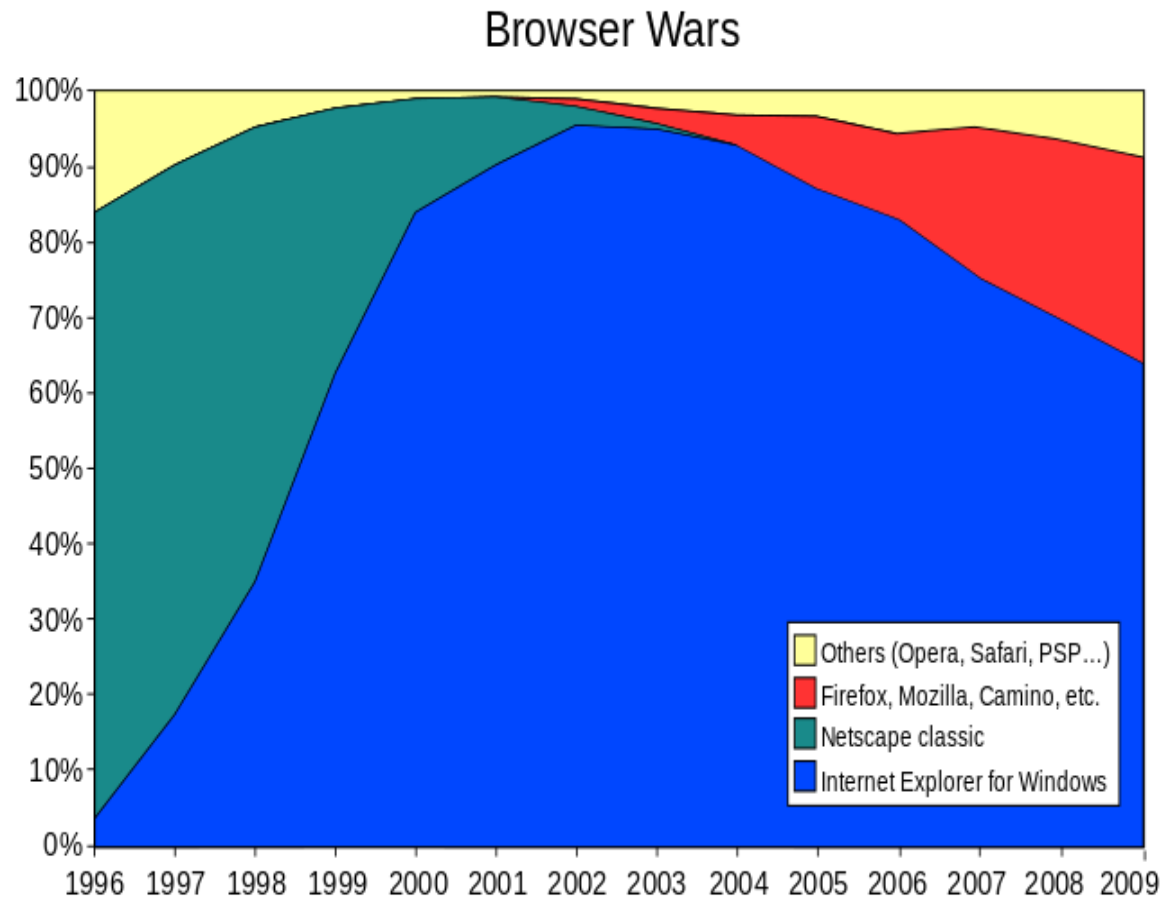
The Future of Search



Развитие WWW



Война браузеров: IE vs Netscape

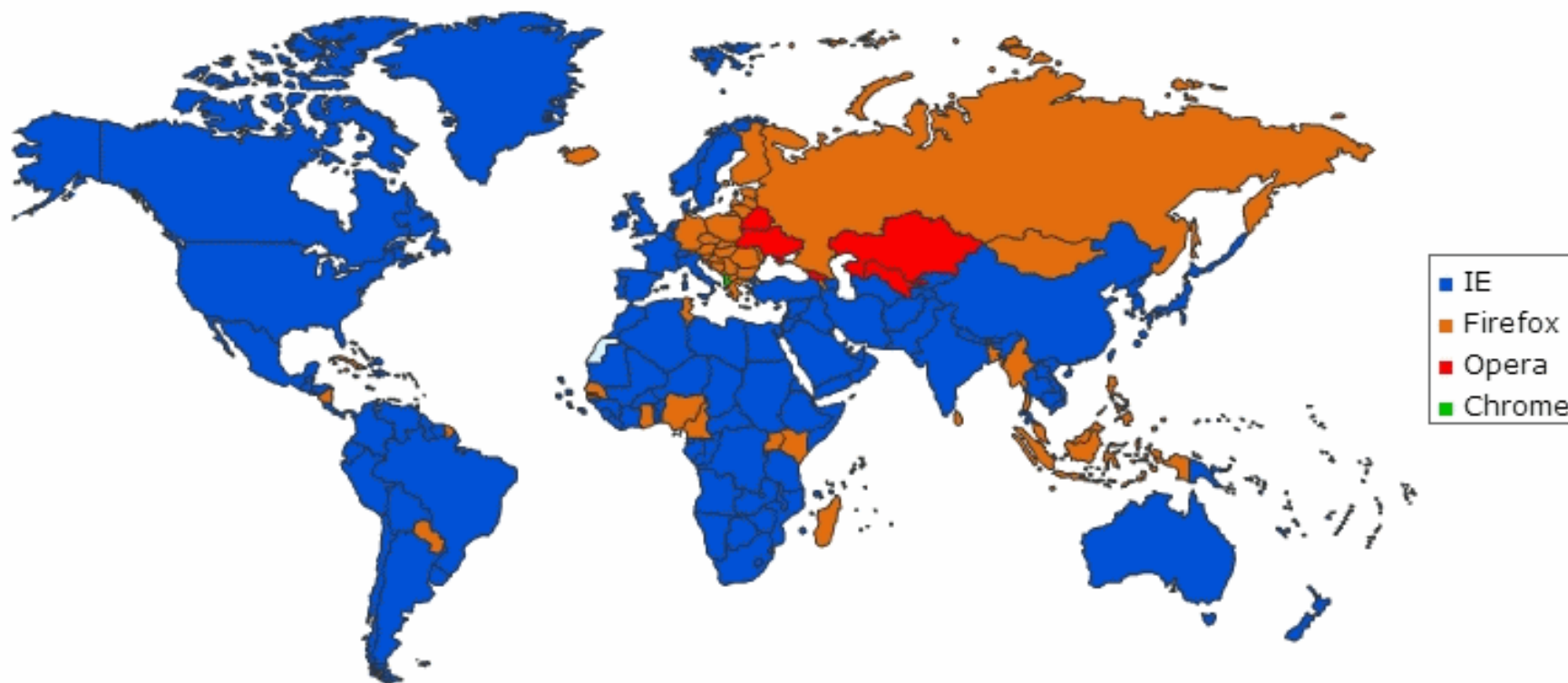


Война браузеров: а потом пришел Chrome

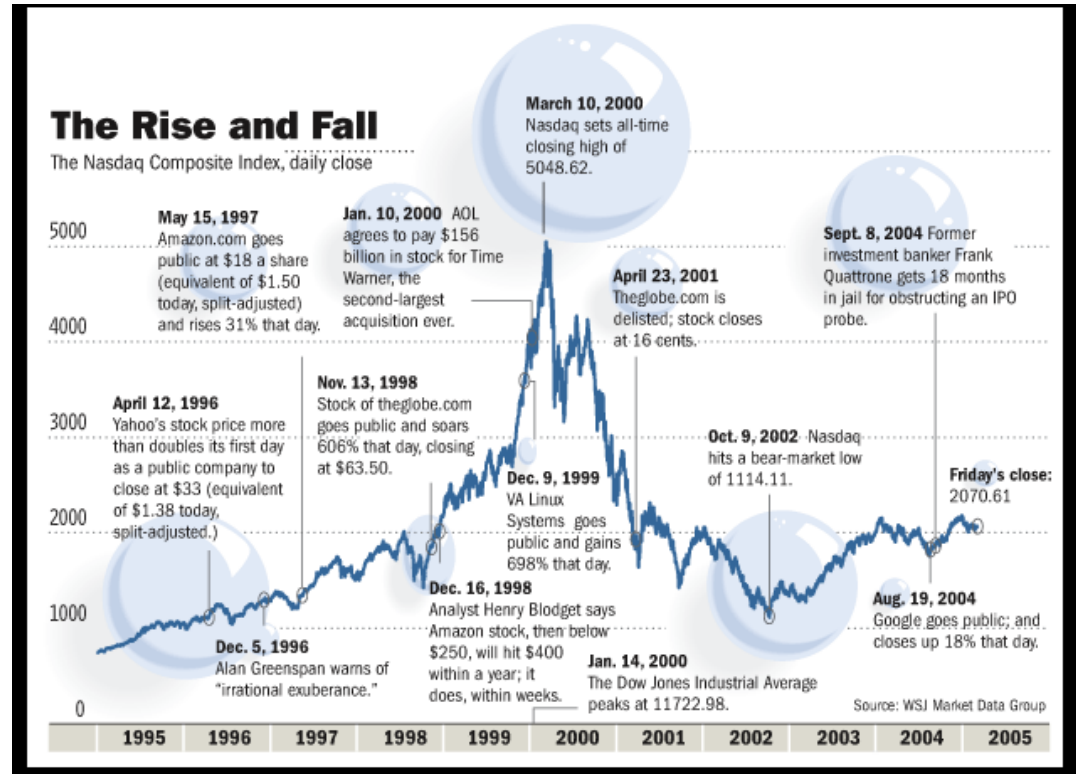


Война браузеров: Chrome vs World

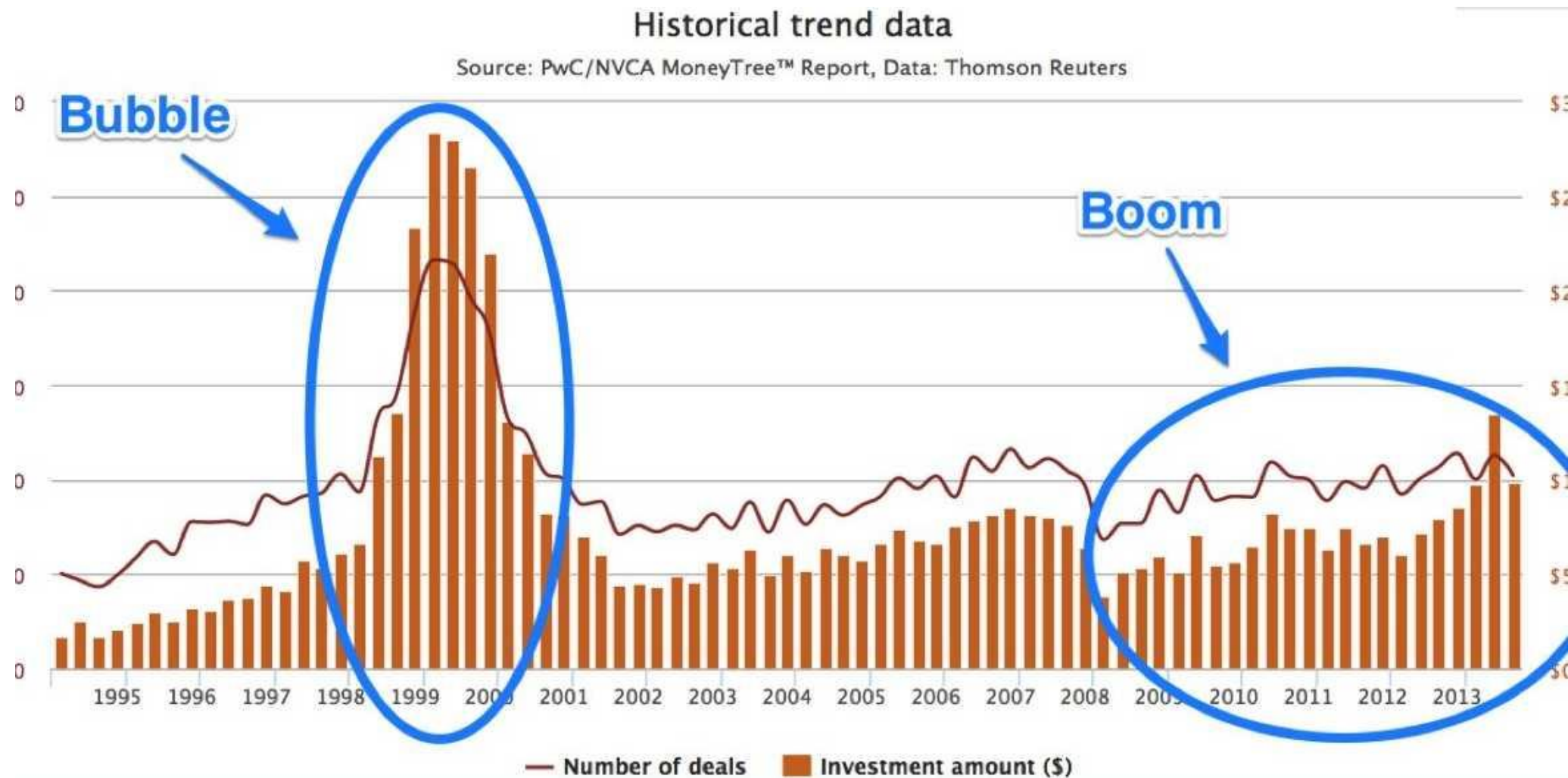
November 2010



Пузырь доткомов (dotcom bubble)



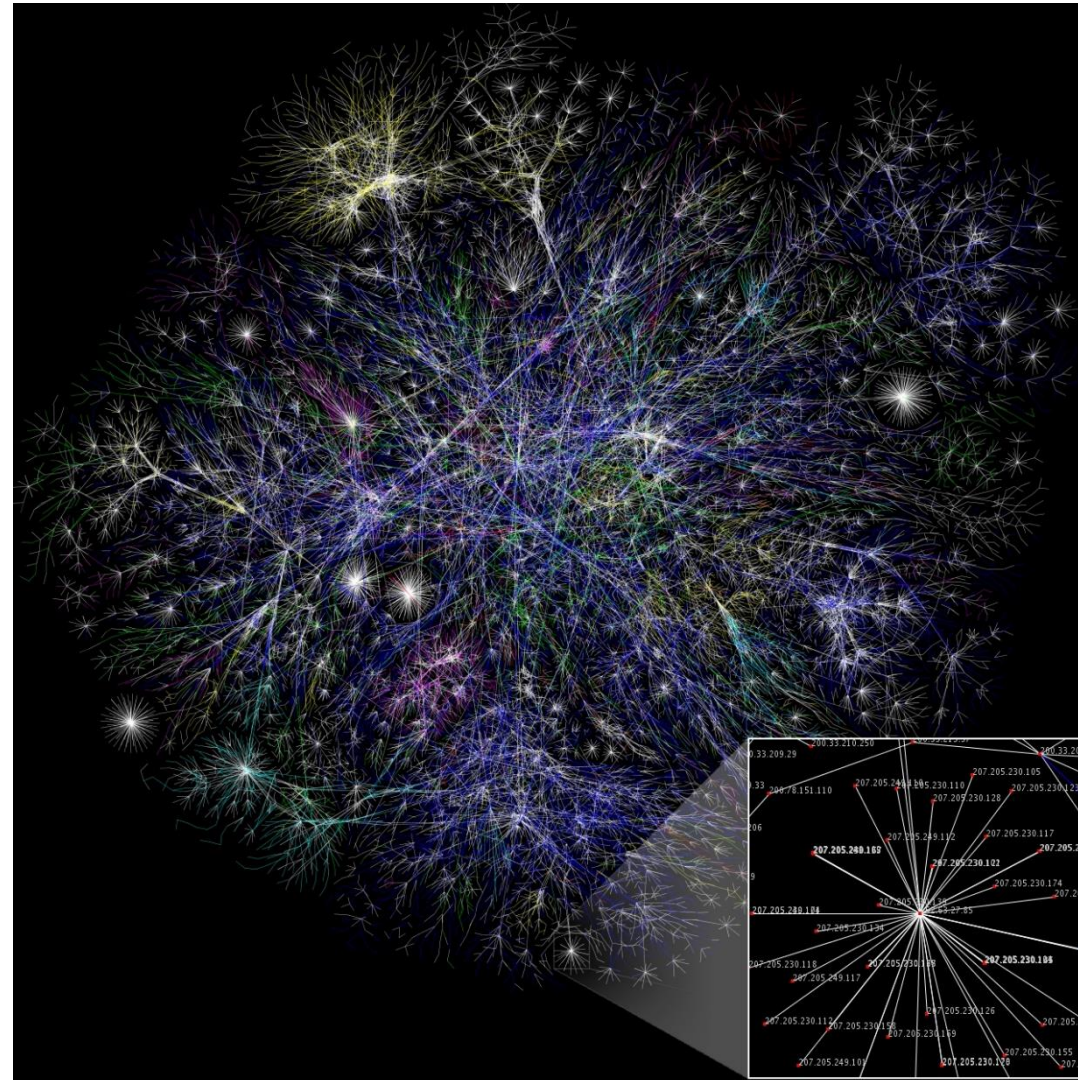
Пузырь доткомов (dotcom bubble)



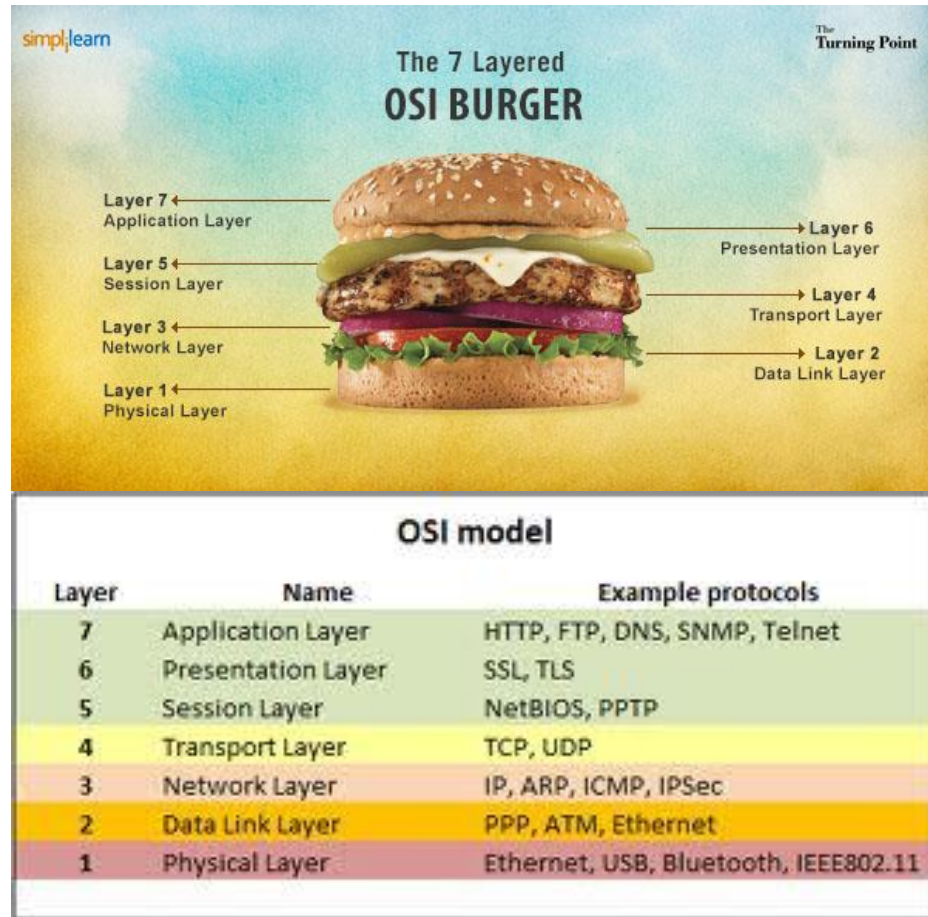
Современное состояние веб-приложений

- **Традиционные сайты:** новости, блоги, wiki, базы знаний, визитки
- **Глобальные приложения:** почтовые сервисы, поиск, социальные сети
- **E-commerce:** магазины, бронирование, цифровая дистрибуция
- **Замена desktop приложениям:** банк-клиенты, CRM, корпоративный софт
- **SAAS**(Software as a Service)

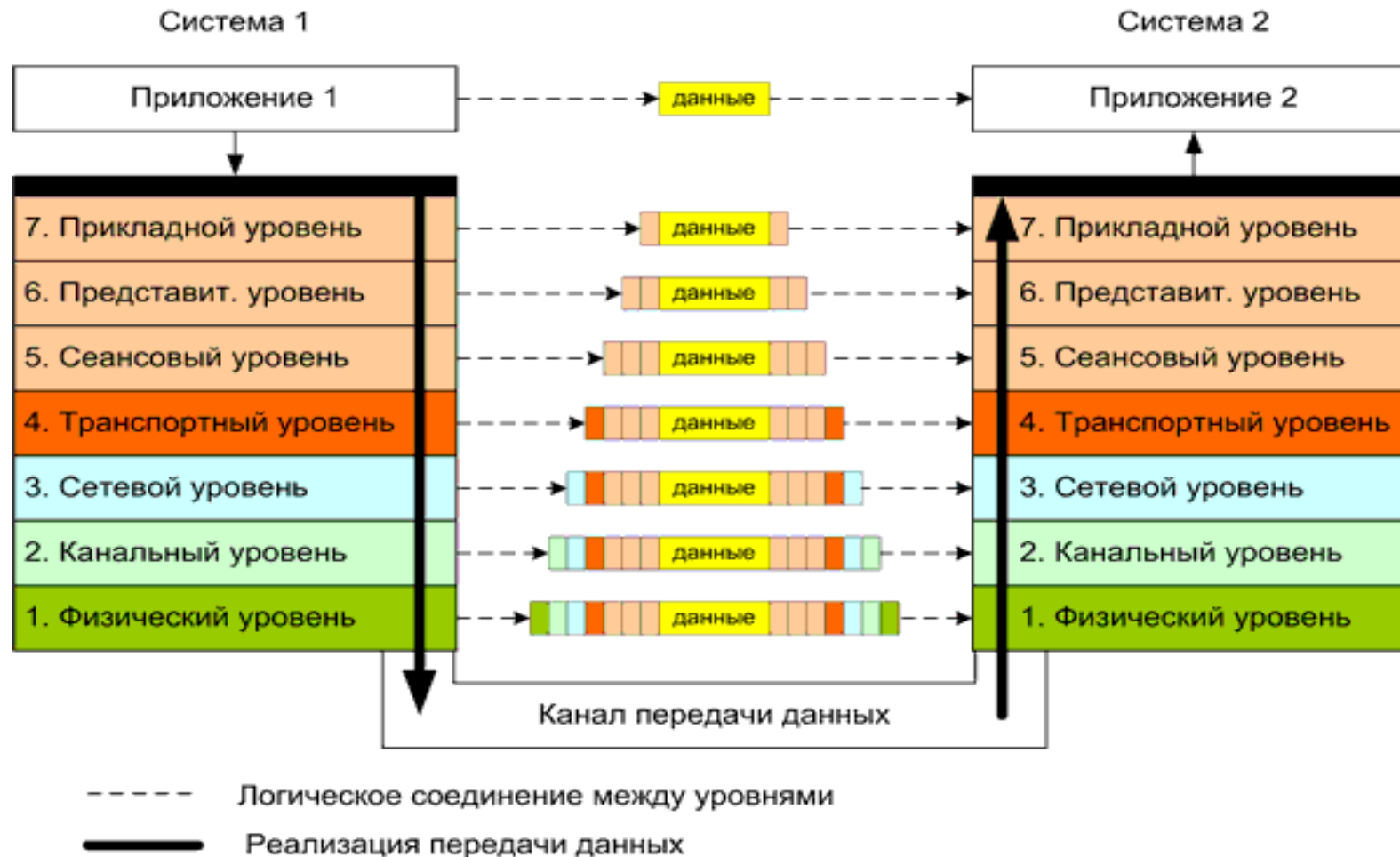
Базовые технологии



OSI ISO vs TCP/IP



OSI ISO

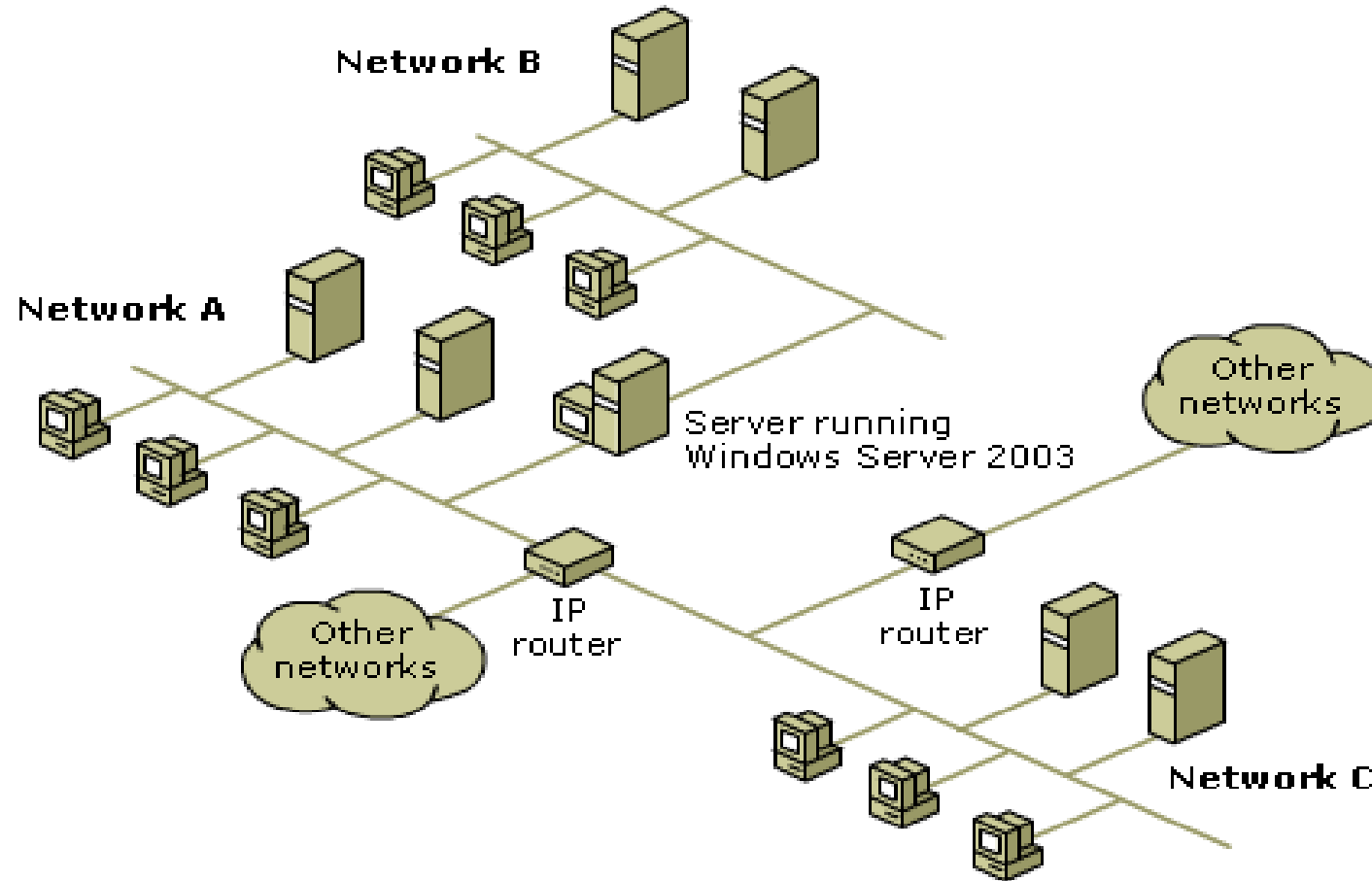


IP: Internet Protocol

- Глобальная адресация
- Передача в гетерогенной сети (сегментация)
- Маршрутизация пакетов



IP: Internet Protocol



IP: Internet Protocol

	Network Portion			Host Portion		
IPv4 Address	192	.	168	.	10	10
	11000000 10101000 00001010				00001010	
Subnet Mask	255	.	255	.	255	0
	11111111 11111111 11111111				00000000	



IP: Internet Protocol

localhost == 127.0.0.1



IP: Адресное пространство ipv4 vs ipv6

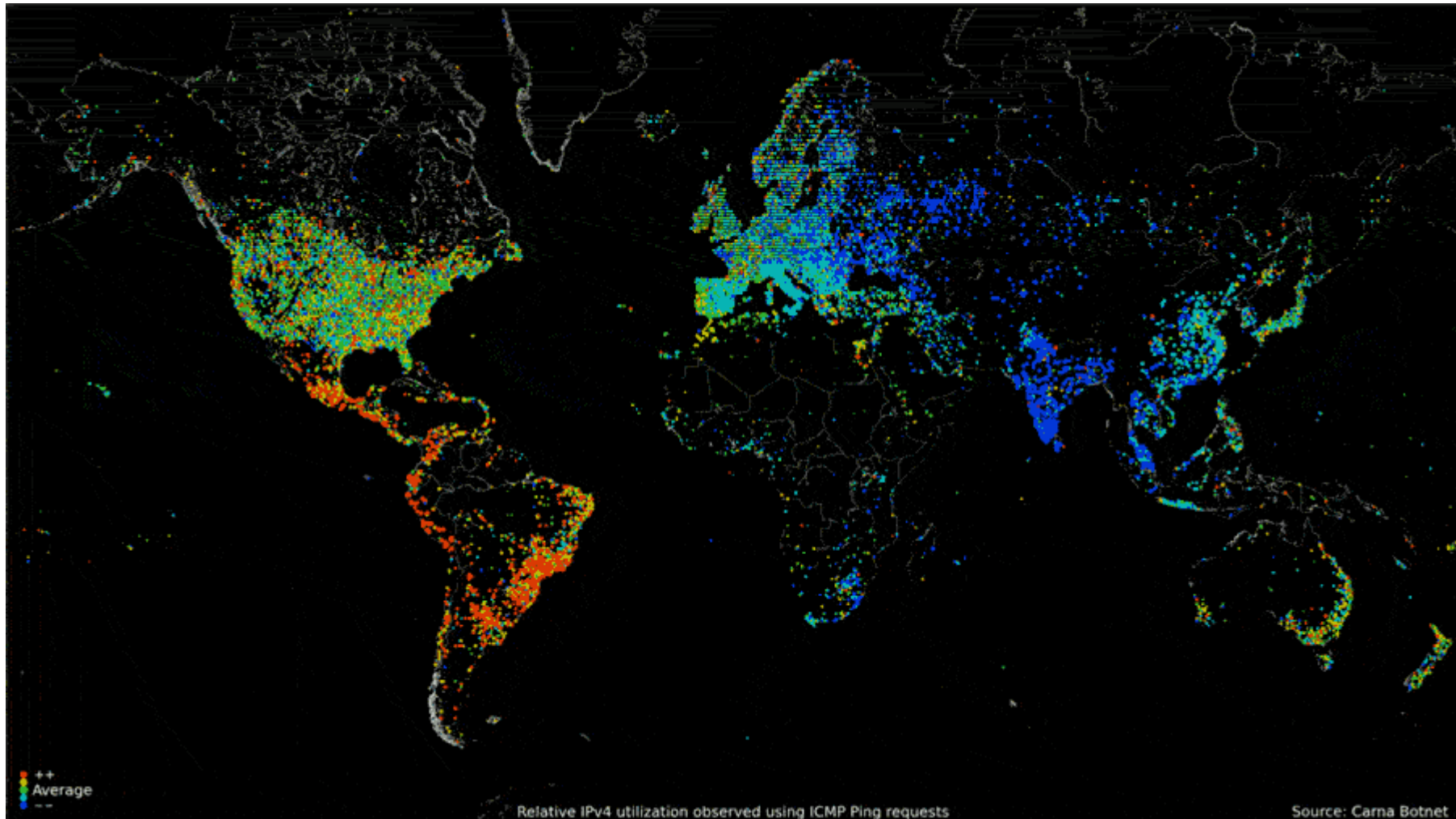
$$2^{32}=4294967295$$

$$2^{128}= 340\ 282\ 366\ 920\ 938\ 463\ 463\ 374\ 607\ 431\ 768\ 211\ 456$$

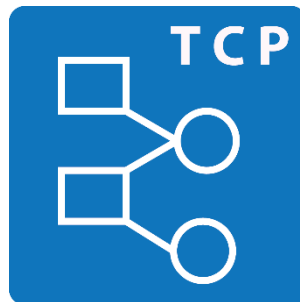
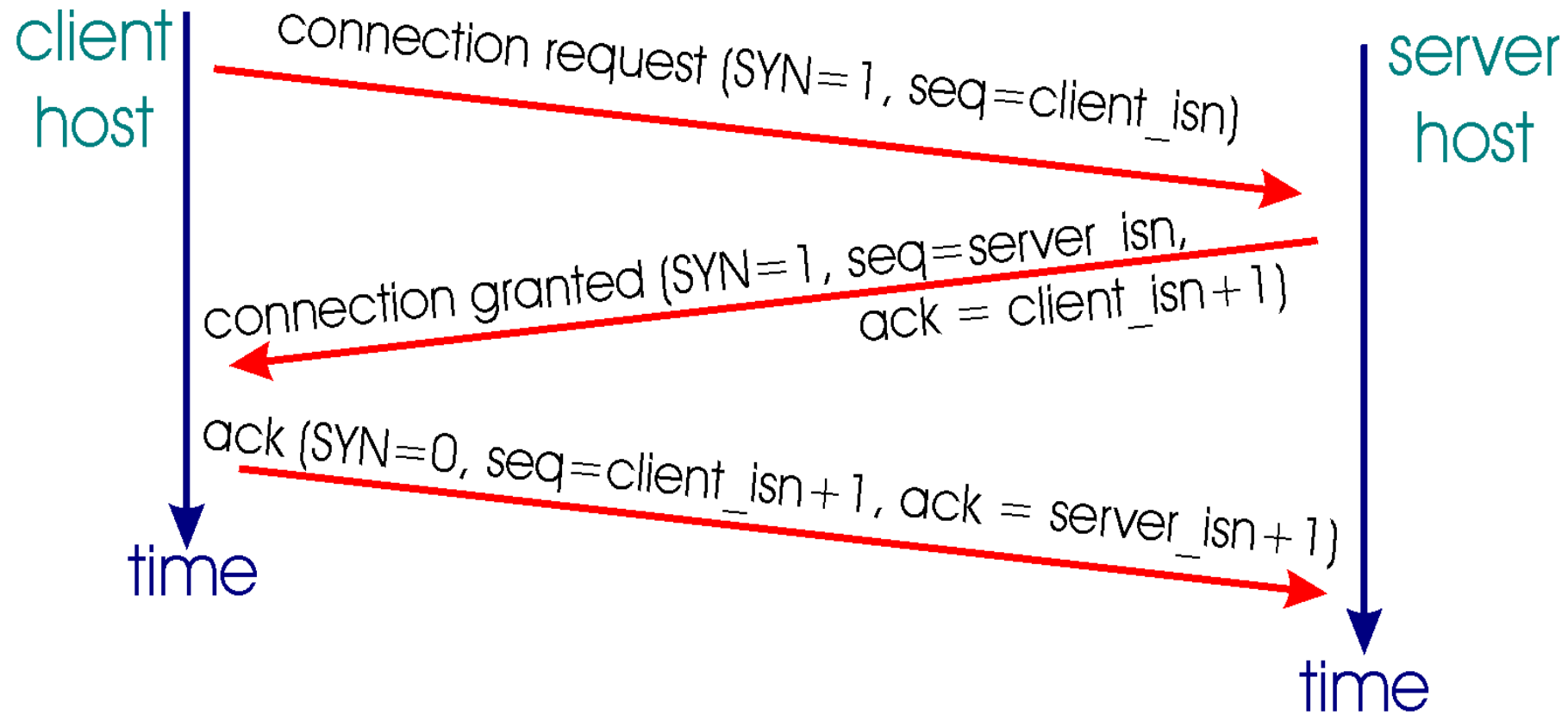


IP

IP: Суточная активность

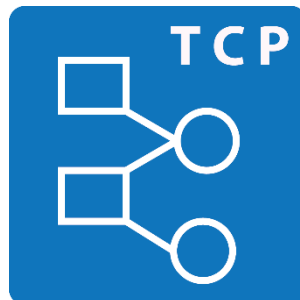


TCP



TCP

- Адресация приложения в пределах хоста с помощью портов
- Последовательное двустороннее соединение
- Надежная доставка
- Управление потоком



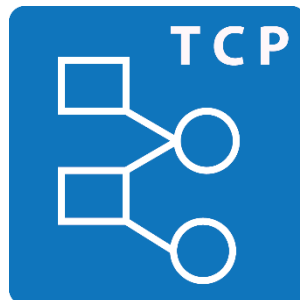
TCP

Порты – адресация приложения на хосте

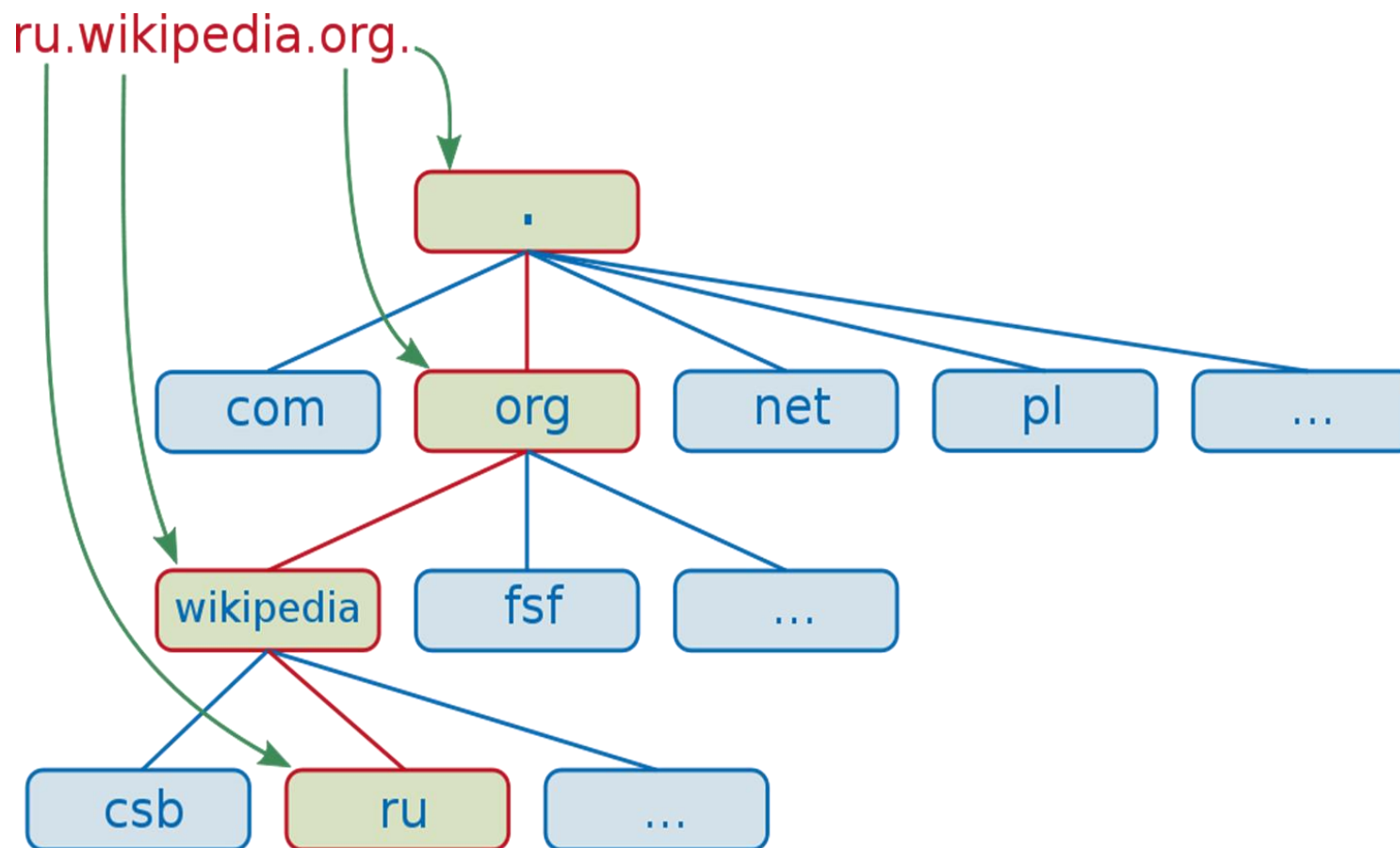
- Well-known: SSH=22, FTP=20,21, HTTP=80, SMTP=25, POP3=110
- Привилегированные (<1024)
- Остальные (>=1024)

Сокеты (sockets) – пара адрес-порт

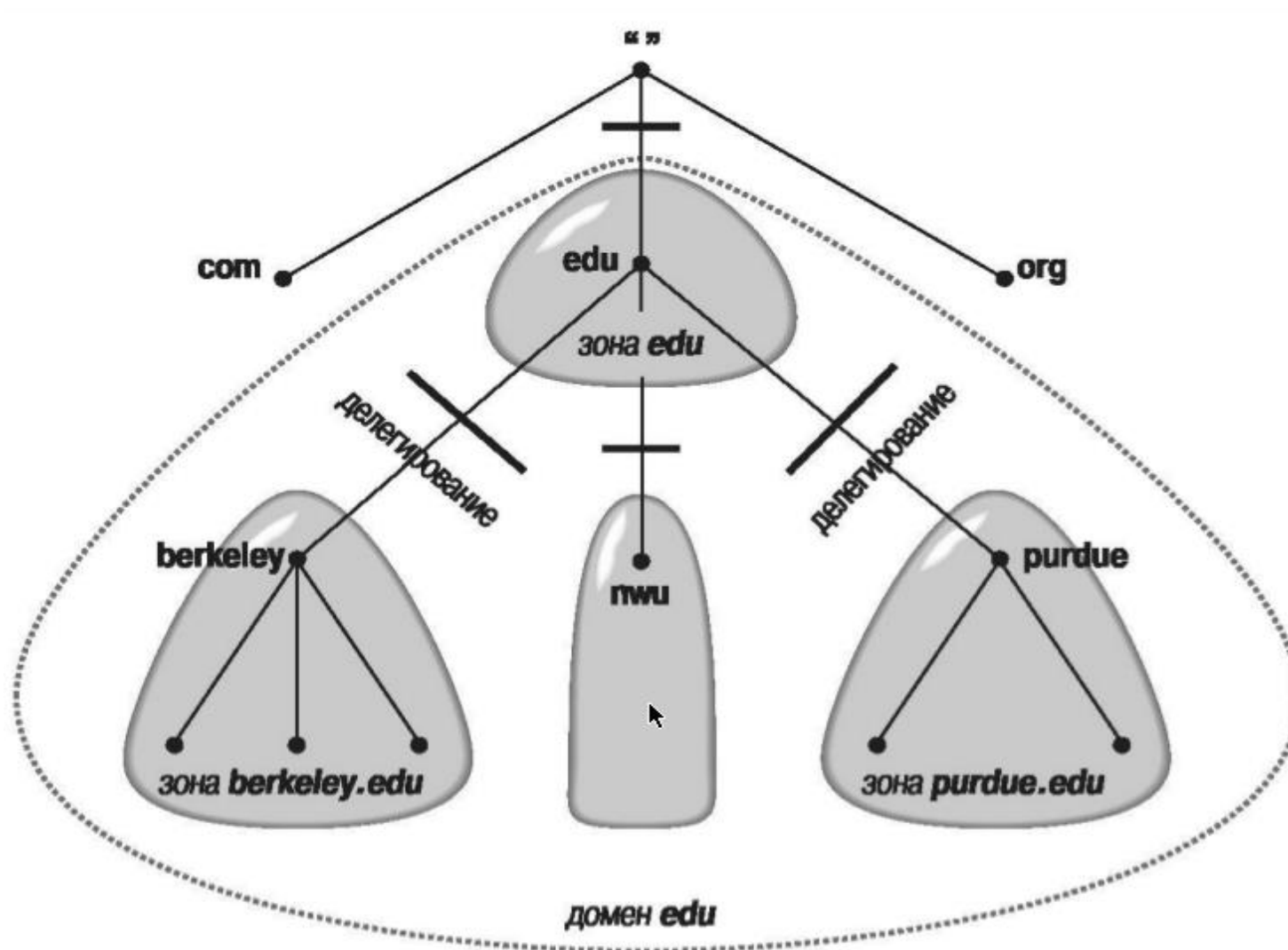
- Серверные (bind, listen, accept)
- Клиентские (connect, send, recv)



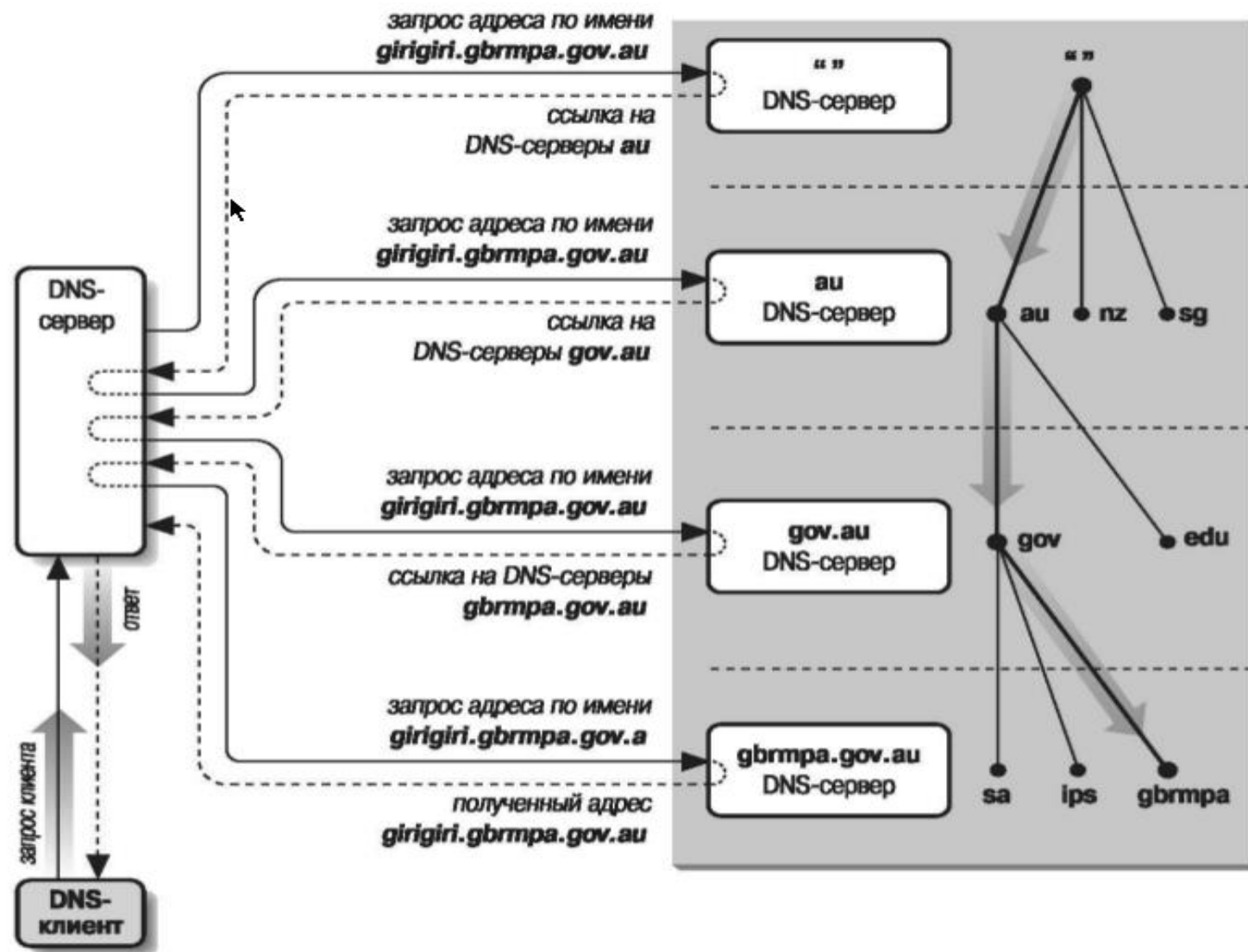
DNS



DNS



DNS



URI == URL || URN || URL + URN

scheme://host:[port]/path/.../[;url-params][?query-string][#anchor]

- **scheme** – протокол (http,https,ftp)
- **host** – ip-адрес или доменное имя
- **port** – порт, необязательно если стандартный
- **path** – путь в файловой системе корневого каталога сервера (или псевдоним, обрабатываемый сервером)
- **url-params** – необязательные пары ключ-значение. Используется для идентификаторов сессии пользователя.
- **query-string** – пары ключ-значение – параметры запроса
- **anchor** – ссылка на позиционный маркер в пределах документа



HTTP: HyperText Transfer Protocol

Текстовый протокол без сохранения состояния

http://

HTTP: Сессия

- Клиент устанавливает TCP-соединение с сервером (обычно, на 80 порт)
- Сервер принимает запрос на соединение и ожидает текст HTTP запроса.
- Клиент отправляет запрос со всей необходимой информацией (URI, тип запроса, заголовки, тело запроса).
- Сервер обрабатывает запрос и отдает ответ: код статуса, заголовки ответа и тело ответа.

HTTP: Структура

Структура HTTP-запроса

- Строка запроса.
- Заголовки.
- Пустая строка.
- Тело запроса.

Структура HTTP-ответа

- Строка статуса ответа.
- Заголовки ответа.
- Пустая строка.
- Тело ответа.

HTTP: Методы

- **OPTIONS** — запрос методов сервера (Allow)
- **GET** — запрос документа (Условный GET)
- **HEAD** — аналог GET, но без тела ответа
- **POST** — передача данных от клиента
- **PUT** — размещение файла по URI/изменение данных
- **DELETE** — удаление файла по URI/удаление данных
- **TRACE, LINK, UNLINK, CONNECT** — редко

HTTP: Коды ответов

- **1xx — Информационные**
- **2xx — Успешное выполнение**
 - 200 — OK
 - 204 — NoContent (только заголовки)
 - 206 — PartialContent (часть ответа)
- **3xx — Перенаправления**
 - 301 — Moved Permanently (SEO, кэширование)
 - 302 — Found (логика работы сайта)
 - 304 — Not Modified (при условном GET)
- **4xx — Ошибка клиента**
 - 400 — Bad Request (размер, формат..)
 - 401 — Unauthorized (запрос авторизации)
 - 403 — Forbidden (allow, deny)
 - 404 — Not Found
 - 408 — Request Timeout (на чтение)
 - 418 — I'm teapot
- **5xx — Ошибка сервера**
 - 500 — Internal Server Error
 - 502 — Bad Gateway (проксирование)
 - 503 — Service Unavailable
 - 504 — Gateway Timeout
 - 505 — HTTP version not supported
 - 507 — Insufficient Storage

http://

HTTP: Заголовки

- **Host** — указание домена, вирт. Хостинг
- **User-Agent** — описание клиента
- **Accept-*** — поддержка MIME типов, кодировок, языков и т.п.
- **Cookie** — куки для данной страницы
- **Referer** — текущая страница
- **If-Modified-Since** — условный GET
- **Connection** — управление соединением
- **Content-Type** — MIME тип документа
- **Content-Length** — размер документа
- **Content-Encoding** — кодирование документа
- **Date** — текущее время сервера
- **Expires** — время актуальности документа
- **Last-Modified** — время изменения файла
- **Set-Cookie** — установка кук для данного URI
- **Connection** — управление соединением

http://

HTTP REST

Resource	GET	PUT	POST	DELETE
Collection Uri, such as http://example.com/resources	List the URIs and perhaps other details of the collection's members.	Replace the entire collection with another collection.	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.	Delete the entire collection.
Element Uri, such as http://example.com/resources/item17	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	Replace the addressed member of the collection, or if it doesn't exist, create it.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it.	Delete the addressed member of the collection.

http://