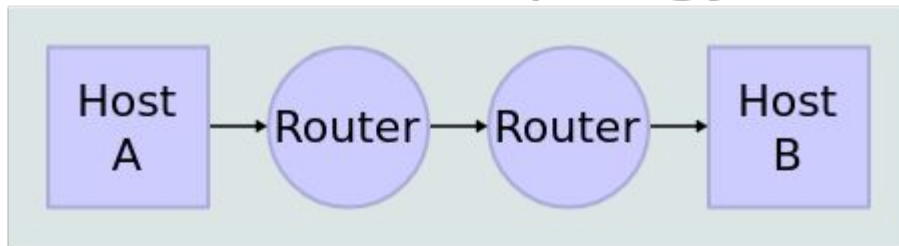


Sistemas Operacionais Embarcados

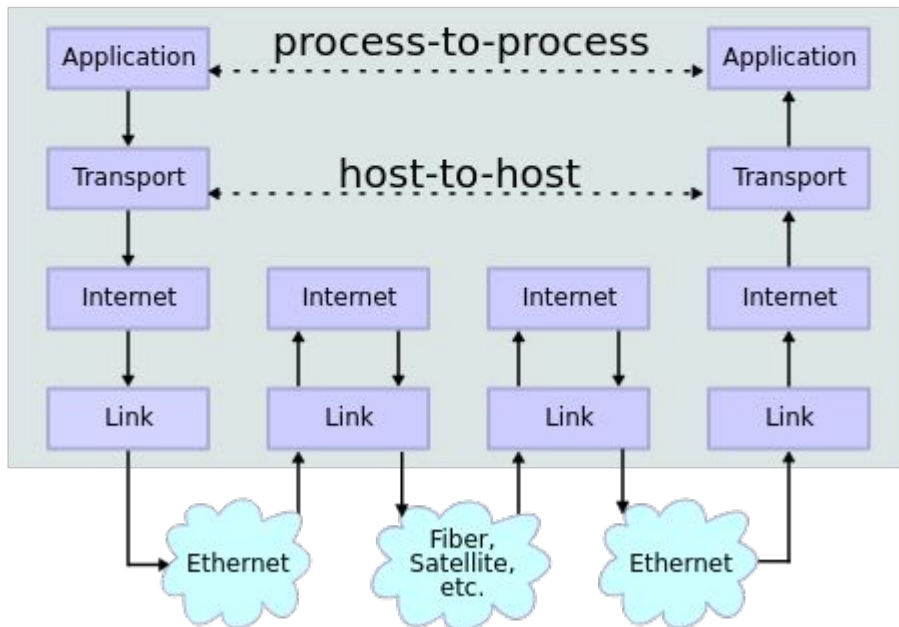
Clientes HTTP

Alguns termos importantes

Network Topology



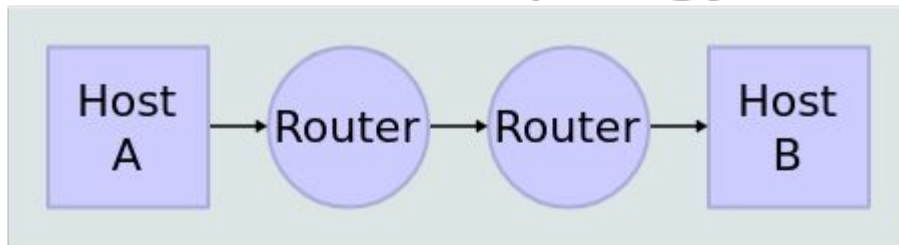
Data Flow



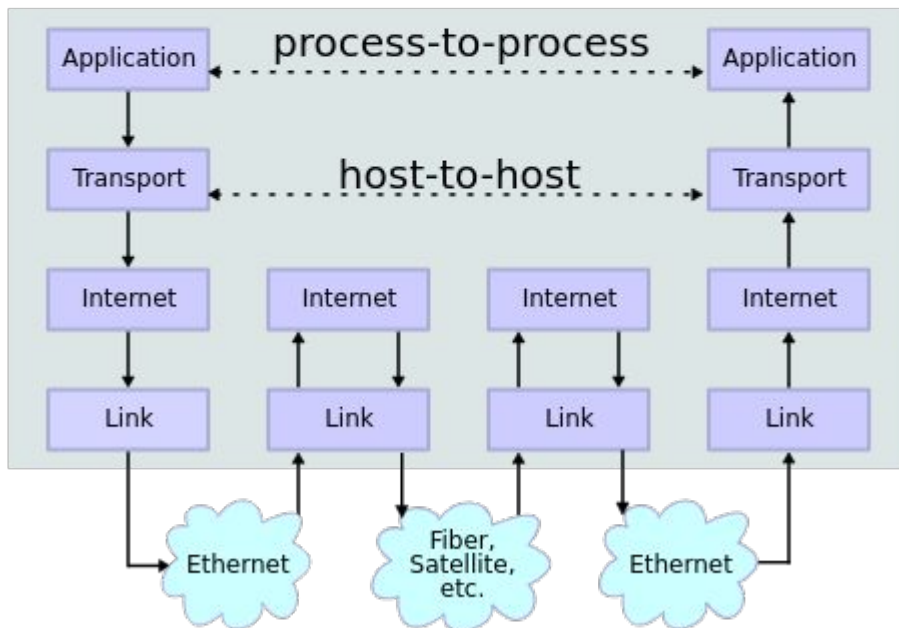
- O TCP/IP é uma *SUÍTE* de protocolos, separada logicamente em 4 camadas:
 - Aplicação
 - Transporte
 - Internet (rede)
 - Link (enlace)

Alguns termos importantes

Network Topology



Data Flow

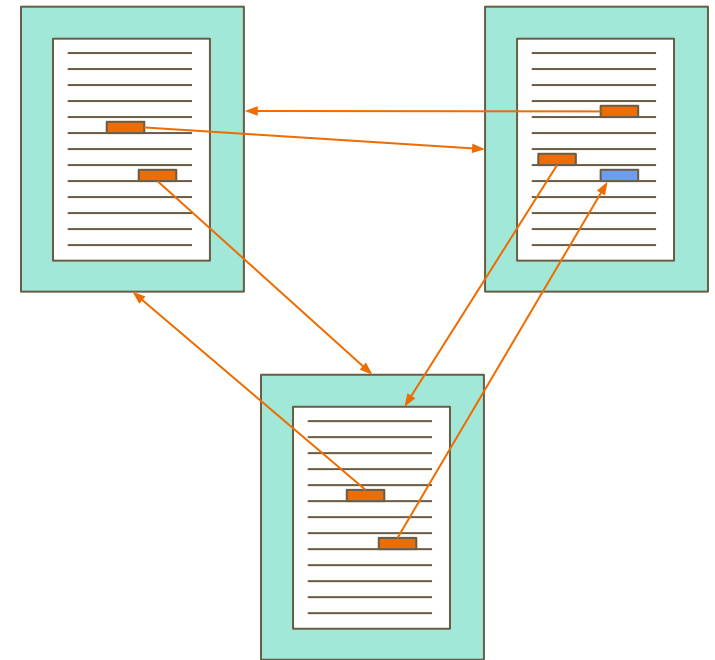


A camada de aplicação representa em alto-nível a troca de informação entre processos. Alguns protocolos possíveis:

- HTTP (web)
- SMTP (email)
- SSH (acesso remoto)

Protocolo HTTP

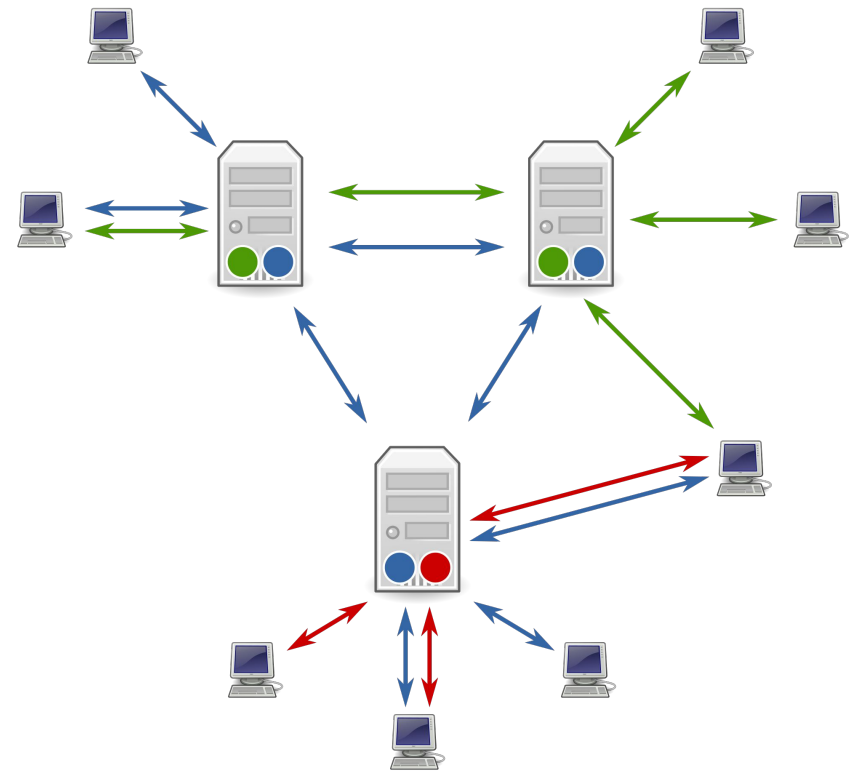
- O *Hypertext Transfer Protocol* (HTTP) é a base da comunicação de dados pela internet
- O termo *hipertexto* se refere a conteúdo (texto, videos, imagens etc.) que contém *ligações* (*links*) a outros conteúdos
- O HTTP é um protocolo do tipo pedido-resposta, dentro do modelo cliente-servidor



Hipertexto

Cenário típico

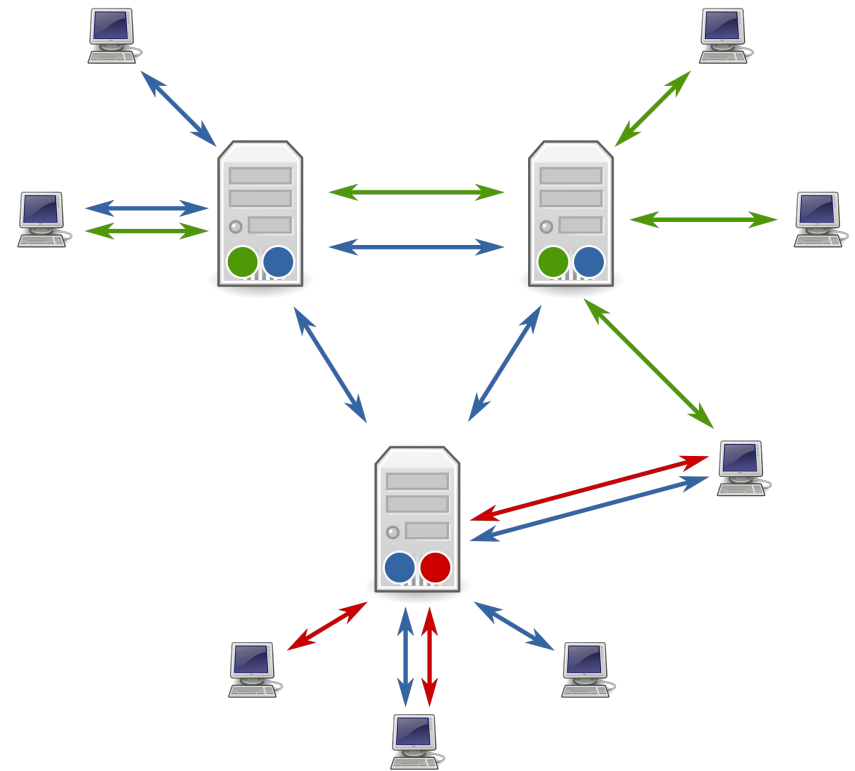
1. O cliente realiza o pedido
2. O servidor que contém a página:
 - a. Separa a informação em **datagramas** (pequenos pedaços independentes), todos com o endereço do cliente como destino
 - b. O servidor manda datagramas para o roteador, que os manda para outro nó mais próximo do cliente (outro roteador ou servidor)



Cenário típico

3. Os datagramas trafegam de servidor a servidor até chegar ao cliente
4. No cliente, os datagramas são agrupados adequadamente e enviados ao programa adequado

Processo semelhante é seguido no envio de emails, em chats, no download de arquivos etc.



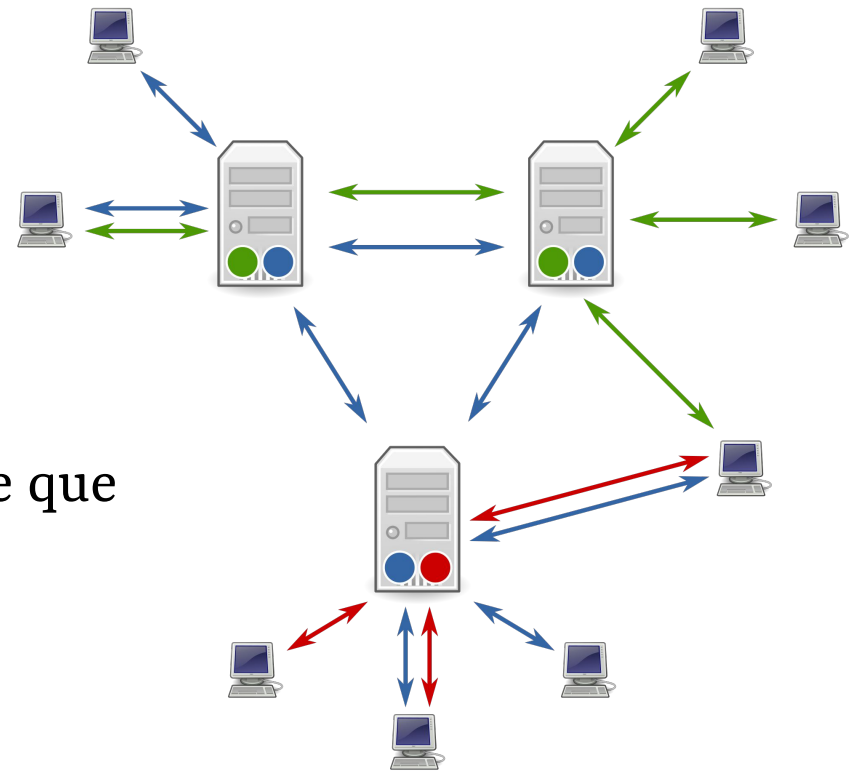
Cenário típico

Do lado do cliente, existem diversos *user agents* que fazem o pedido em nome do usuário:

- browsers
- aplicativos de celular
- web crawlers
- sistemas embarcados etc

Existem também diversos elementos de rede que melhoram a comunicação entre clientes e servidores:

- web cache
- servidores de upstream
- servidores de proxy



Cenário típico

- Para acessarmos recursos no servidor (páginas HTML, documentos etc.), usamos a URL (*Uniform Resource Locator*):

`protocolo://domínio:porta/caminho/recurso?query_string#fragmento`

Exemplos:

`http://www.w3.org/Addressing/URL/uri-spec.html`

`https://www.youtube.com/watch?v=PpsEaqJV_A0&t=153`

Cenário típico

- Para não termos de decorar o IP do servidor, usamos o DNS (*Domain Name Structure*), que traduz os hostnames em endereços IP
- Exemplo: vá em <https://www.whatismyip.com/dns-lookup/>, e digite www.google.com na barra de busca
- Neste exemplo, encontramos o endereço de IP 172.217.8.4
- Digite 172.217.8.4 em um browser e veja o resultado

DNS Lookup

Ad closed by Google

[Report this ad](#) [Why this ad? ⓘ](#)

URL:

IPv4 address for www.google.com
Domain Name Server: 172.217.8.4

Use the DNS lookup tool to find the IP address of a certain domain name. The results will include the IP addresses in the DNS records received from the name servers.

```

#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

Code/11_Clientes_HTTP/01_http_client.c

```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);

```

(Continuação)

```

#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sock
    int socke
    int port
    char *hos
        buf[E
    FILE *fp

    host = arg
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

- Este programa cria um cliente que baixa uma página da internet, salvando-a no arquivo "saida.html"
- Modo de Uso:
./01_http_client.out HOSTNAME PAGINA

```

socket_id = socket(AF_INET,
    SOCK_STREAM, IPPROTO_TCP);
ip = get_ip(host);
memset(&servidorAddr, 0,
    sizeof(servidorAddr));
servidorAddr.sin_family =
    AF_INET;
servidorAddr.sin_addr.s_addr =
    inet_addr(ip);
servidorAddr.sin_port =

    *)
    f,
    addr)) < 0)

    exao!\n");

    exit(-4);
}
write(socket_id, get,
    strlen(get));
free(get);
free(ip);

```

```
#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);
```

**Todos os headers
necessários foram incluídos
neste arquivo**

```
socket(AF_INET,
        IPPROTO_TCP);
    );
    servAddr, 0,
    servidorAddr));
    in_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);
```

Code/11_Clientes_HTTP/01_http_client.c

(Continuação)

```

#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

Se o usuário não indicar o endereço e a página desejada, não é possível fazer o pedido

```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    port = atoi(argv[2]);
    servidorAddr.sin_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);

```

```

#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

Code/11_Clientes_HTTP/01_http_client.c

```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,

```

**A função `build_get_query()`,
definida em “client_func.c”,
criará uma string que pede a
página “page” no servidor
“host” usando o protocolo
HTTP**

(Continuação)

```

#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));

```

Aqui veremos o pedido da página

Criamos um socket

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main(int argc, char *argv[])
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);
```

```
    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
        sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);
```

Code/11_Clientes_HTTP/01_http_client.c

(Continuação)

Pegamos o IP do servidor usando DNS



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);
```

```
    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);
```

Code/11_Clientes_HTTP/01_http_client.c

(Continuação)

```

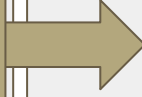
#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int ip;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get_query(host, page);
    fprintf(stderr,
        "Pedido HTTP:\n");
    fprintf(stderr, "%s", get);

```

Abrimos uma conexão com o servidor



```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);

```

Code/11_Clientes_HTTP/01_http_client.c

(Continuação)

```

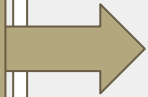
#include "client_funcs.h"

int main(int argc, char **argv)
{
    if(argc != 3)
    {
        fprintf(stderr, "Uso: %s host
pagina\n", argv[0]);
        exit(-1);
    }
    struct sockaddr_in servidorAddr;
    int socket_id;
    int port = 80;
    char *host, *ip, *get, *page,
        buf[BUFSIZ+1];
    FILE *fp;

    host = argv[1];
    page = argv[2];
    get = build_get(page);
    fprintf(stderr, "Pedido: %s\n", get);
    fprintf(stderr, "%s", get);

```

**Mandamos o pedido
HTTP da página**



```

    socket_id = socket(AF_INET,
        SOCK_STREAM, IPPROTO_TCP);
    ip = get_ip(host);
    memset(&servidorAddr, 0,
        sizeof(servidorAddr));
    servidorAddr.sin_family =
        AF_INET;
    servidorAddr.sin_addr.s_addr =
        inet_addr(ip);
    servidorAddr.sin_port =
        htons(port);
    if(connect(socket_id,
        (struct sockaddr *)
            &servidorAddr,
            sizeof(servidorAddr)) < 0)
    {
        fprintf(stderr,
            "Erro na conexao!\n");
        exit(-4);
    }
    write(socket_id, get,
        strlen(get));
    free(get);
    free(ip);

```

```

    fprintf(stderr, "Recebendo o
resultado HTML e o escrevendo no
arquivo 'saida.html'...\n");
    fp = fopen("saida.html", "w");
    int tmpres;
    while((tmpres = read(socket_id,
        buf, BUFSIZ)) > 0)
    {
        buf[tmpres] = '\0';
        fprintf(fp, "%s", buf);
    }
    if(tmpres < 0)
        fprintf(stderr, "Erro no
recebimento de dados!\n");
    close(socket_id);
    fclose(fp);
    return 0;
}

```

Code/11_Clientes_HTTP/01_http_client.c
(continuação)

```

#ifndef CLIENT_FUNCS_H
#define CLIENT_FUNCS_H

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>

char *build_get_query(char *host,
    char *page);
char *get_ip(char *host);

#endif // CLIENT_FUNCS_H

```

Code/11_Clientes_HTTP/client_funcs.h

```

    fprintf(stderr, "Recebendo o
    resultado HTML e o escrevendo no
    arquivo 'saida.html'...\n");
    fp = fopen("saida.html", "w");
    int tmpres;
    while((tmpres = read(socket_id,
        buf, BUFSIZ)) > 0)
    {
        buf[tmpres] = '\0';
        fprintf(fp, "%s", buf);
    }
    if(tmpres < 0)
        fprintf(stderr, "Erro no
        recebimento de dados!\n");
    close(socket_id);
    fclose(fp);
    return 0;
}

```

Code/11_Clientes_HTTP/01_http_client.c
(continuação)

```

#ifndef CLIENT_FUNCS_H
#define CLIENT_FUNCS_H

#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>

char *build_get_query(char *host,
    char *page);
char *get_ip(char *host);

#endif // CLIENT_FUNCS_H

```

Code/11_Clientes_HTTP/client_funcs.h

**Abrimos o arquivo aonde vamos
escrever a página recebida**

```

    fprintf(stderr, "Recebendo o
    resultado HTML e o escrevendo no
    arquivo 'saida.html'...\n");
    fp = fopen("saida.html", "w");
    int tmpres;
    while((tmpres = read(socket_id,
        buf, BUFSIZ)) > 0)
    {
        buf[tmpres] = '\0';
        fprintf(fp, "%s", buf);
    }
    if(tmpres < 0)
        fprintf(stderr, "Erro no
        recebimento de dados!\n");
    close(socket_id);
    fclose(fp);
    return 0;
}

```

Code/11_Clientes_HTTP/01_http_client.c
(continuação)

```

#ifndef CLIENT_FUNCS_H
#define CLIENT_FUNCS_H

```

```

#include
#include
#include
#include

```

```

#include <stdlib.h>
#include <netdb.h>
#include <string.h>

```

```

char *build_get_query(char *host,
    char *page);
char *get_ip(char *host);

```

```

#endif // CLIENT_FUNCS_H

```

Code/11_Clientes_HTTP/client_funcs.h

**Vamos lendo o que o servidor
manda, de BUFSIZ em BUFSIZ
bytes (valor definido em
"stdio.h")...**

```

    fprintf(stderr, "Recebendo o
    resultado HTML e o escrevendo no
    arquivo 'saida.html'...\n");
    fp = fopen("saida.html", "w");
    int tmpres;
    while((tmpres = read(socket_id,
        buf, BUFSIZ)) > 0)
    {
        buf[tmpres] = '\0';
        fprintf(fp, "%s", buf);
    }
    if(tmpres < 0)
        fprintf(stderr, "Erro no
        recebimento de dados!\n");
    close(socket_id);
    fclose(fp);
    return 0;
}

```

Code/11_Clientes_HTTP/01_http_client.c
(continuação)

```

#ifndef CLIENT_FUNCS_H
#define CLIENT_FUNCS_H

```

```

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

```

```

#i
#
#
#
#
#

```

**... e escrevendo os bytes
recebidos em "saida.html"**

```

char *build_get_query(char *host,
    char *page);
char *get_ip(char *host);

```

```

#endif // CLIENT_FUNCS_H

```

Code/11_Clientes_HTTP/client_funcs.h

```

    fprintf(stderr, "Recebendo o
    resultado HTML e o escrevendo no
    arquivo 'saida.html'...\n");
    fp = fopen("saida.html", "w");
    int tmpres;
    while((tmpres = read(socket_id,
        buf, BUFSIZ)) > 0)
    {
        buf[tmpres] = '\0';
        fprintf(fp, "%s", buf);
    }
    if(tmpres < 0)
        fprintf(stderr, "
recebimento de dados!\n");
    close(socket_id);
    fclose(fp);
    return 0;
}

```

**Finalmente, fechamos o socket e
o arquivo "saida.html"**

```

#ifndef CLIENT_FUNCS_H
#define CLIENT_FUNCS_H

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>

```

```

get_query(char *host,
    ge);
(char *host);
CLIENT_FUNCS_H

```

Code/11_Clientes_HTTP/01_http_client.c
(continuação)

Code/11_Clientes_HTTP/client_funcs.h


```
#include "client_funcs.h"

char *build_get_query(char *host,
    char *page)
{
    char *query;
    char *getpage = page;
    query = (char *)
        malloc(100+ strlen(host) +
            strlen(getpage));
    sprintf(query, "GET %s
HTTP/1.1\r\nHost: %s\r\nUser-Agent:
HTMLGET 1.1\r\nAccept: */*\r\n\r\n",
        getpage,
        host);
    return query;
}
```

Code/11_Clientes_HTTP/client_funcs.c

```
char *get_ip(char *host)
{
    struct hostent *hent;
    int iplen = 15;
    //XXX.XXX.XXX.XXX
    char *ip = (char *)
        malloc(iplen+1);
    memset(ip, 0, iplen+1);
    hent = gethostbyname(host);
    if(hent == NULL)
    {
        perror("Não foi possível
obter o IP");
        exit(-2);
    }
    if(inet_ntop(AF_INET, (void*)
        hent->h_addr_list[0],
        ip, iplen) == NULL)
    {
        perror("Can't resolve
host");
        exit(-3);
    }
    return ip;
}
```

(Continuação)

```
#include "client_funcs.h"

char *build_get_query(char *host,
    char *page)
{
    char *query;
    char *getpage = page;
    query = (char *)
        malloc(100+ strlen(host) +
            strlen(getpage));
    sprintf(query, "GET %s
HTTP/1.1\r\nHost: %s\r\nUser-Agent:
HTMLGET 1.1\r\nAccept: */*\r\n\r\n",
        getpage,
        host);
    return query;
}
```

Code/11_Clientes_HTTP/client_funcs.c

```
char *get_ip(char *host)
{
    struct hostent *hent;
    int iplen = 15;
    //XXX.XXX.XXX.XXX
    char *ip = (char *)
```

Aqui criamos o pedido da página usando o protocolo HTTP

Por exemplo, para baixar a página <https://www.noticias.unb.br/publicacoes>, o pedido HTTP é

```
GET /publicacoes HTTP/1.1
Host: www.noticias.unb.br
User-Agent: HTMLGET 1.1
Accept: */*
```

(Ou seja, enviamos esse texto para o servidor, que responde enviando a página)

```
}
```

(Continuação)

Esta função as bibliotecas de sockets e de acesso à internet para traduzir o *hostname* em um endereço de IP (DNS)

```
#include "cl...  
  
char *build_  
char *pa_  
{  
    char *qu_  
    char *getpage = page;  
    query = (char *)  
        malloc(100+ strlen(host) +  
            strlen(getpage));  
    sprintf(query, "GET %s  
HTTP/1.1\r\nHost: %s\r\nUser-Agent:  
HTMLGET 1.1\r\nAccept: */*\r\n\r\n",  
        getpage,  
        host);  
    return query;  
}
```

Code/11_Clientes_HTTP/client_funcs.c

```
char *get_ip(char *host)  
{  
    struct hostent *hent;  
    int iplen = 15;  
    //XXX.XXX.XXX.XXX  
    char *ip = (char *)  
        malloc(iplen+1);  
    memset(ip, 0, iplen+1);  
    hent = gethostbyname(host);  
    if(hent == NULL)  
    {  
        perror("Não foi possível  
obter o IP");  
        exit(-2);  
    }  
    if(inet_ntop(AF_INET, (void*)  
        hent->h_addr_list[0],  
        ip, iplen) == NULL)  
    {  
        perror("Can't resolve  
host");  
        exit(-3);  
    }  
    return ip;  
}
```

(Continuação)