Resolução de Nomes e Endereços

MC833 – Programação em Redes de Computadores
Instituto de Computação – UNICAMP

Carlos A. Astudillo Trujillo

Roteiro

1. Noção sobre nomes e funcionamento do DNS
2. Funções para nomes de máquinas
3. Funções para nomes de serviços
4. Funções auxiliares

A maior parte do conteúdo é do Capitulo 11 do livro texto.

Conversão de nomes

- Endereço IP (host) e portas (serviços) são números
- Ideal é usar nomes
- Hosts
  - *Simple Name* – **myserver**
  - *Fully Qualified Domain Name* (FQDN) ou *Absolute Name* – **myserver.unicamp.br**

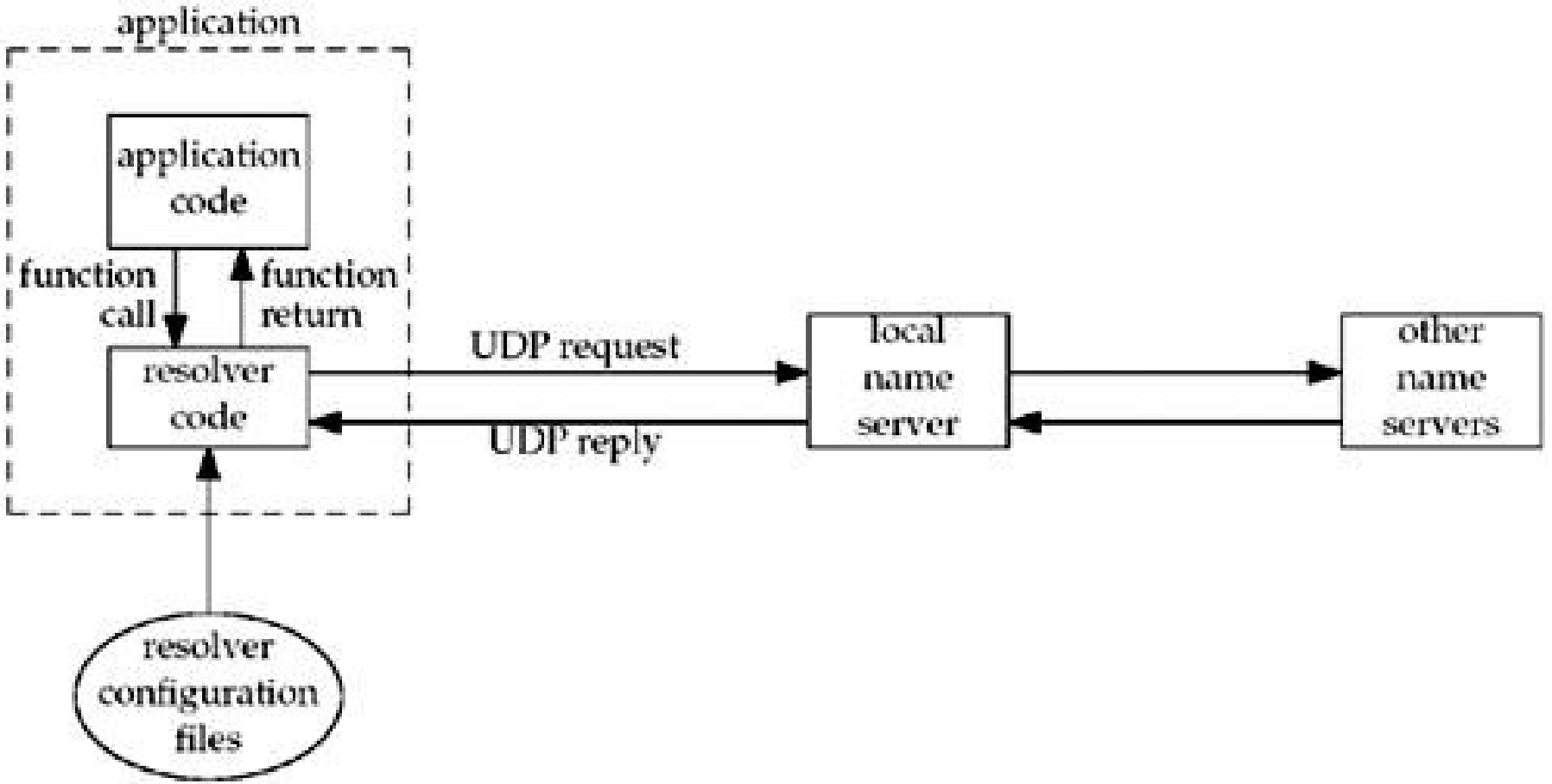
Domain Name System

- Funciona como uma Lista Telefônica.
- É um sistema hierárquico e distribuído de nomes para computadores, serviços ou qualquer outro recurso ligado à Internet.
- Traduz Nomes de Domínio para Endereços IP que são requeridos para localizar serviços de computador e dispositivos.
- É um componente essencial do funcionamento da Internet.

Domain Name System

- É um serviço cliente/servidor baseado em hierarquias
- Parte cliente é conhecida como *resolver* e a parte servidora como *name server*.
- As conexões entre o cliente e o servidor se dão através do protocolo UDP e porta 53.

Domain Name System



Domain Name System

- *O resolver* contata o servidor DNS por chamadas *resolvers*:

  `gethostbyname` e **`gethostbyaddr`**

- Servidor geralmente é implementado em linux usando o software *Berkeley Internet Name Domain* (BIND).

Arquivo de Configuracao no Servidor DNS (BIND)

```
// Boot file for unicamp.br name server
// Exemplo de conteudo do arquivo /etc/named.conf no servidor DNS

options {
        directory "/var/named";
};

…

zone "unicamp.br" {
        type master;
        file "zone/unicamp.br";
};
```

Resource Records (RR)

- RR são as entradas no *Domain Name Server* (DNS).

```
#Conteudo arquivo /var/named/zone/unicamp.br
$ORIGIN unicamp.br  ; start of this zone file in the namespace
$TTL 1h             ; default expiration time of all resource
                                records without their own TTL value
localhost     IN     A      127.0.0.1
mail1         IN     A      192.168.0.101
mail12        IN     A      192.168.0.102
myserver      IN     A      192.168.0.100
              IN     AAAA   3ffe:b80:1f8d:1:a00:20ff:fea7:686b
              IN     MX     5 mail1.unicamp.br.
              IN     MX     10 mail2.unicamp.br.
ftp           IN     CNAME  myserver.unicamp.br
www           IN     CNAME  myserver.unicamp.br
funny         IN     A      192.168.0.100
```

Resource Records (RR)

- **`A`** – nome para IPv4.
- **`AAAA`** (“quad A”) – nome para Ipv6.
- **`PTR`** – endereço IP em nome de estação – cada byte do endereço IPv4 é convertido para decimal e então para ASCII.
- **`MX`** – mail exchanger para a estação.
- **`CNAME`** – usado para associar registros a nome de serviços comuns (nome em nome).

O dilema da galinha e o ovo na configuracao do servidor DNS

Alternativas ao DNS

- Arquivos estáticos (**`/etc/hosts`**).
- Network Information Service (NIS).
- Particular para cada SO.

Exemplo do arquivo /etc/hosts

```
castudillo@quechua:~$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    quechua

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Função gethostbyname

- Os host são conhecidos geralmente por nomes legíveis.

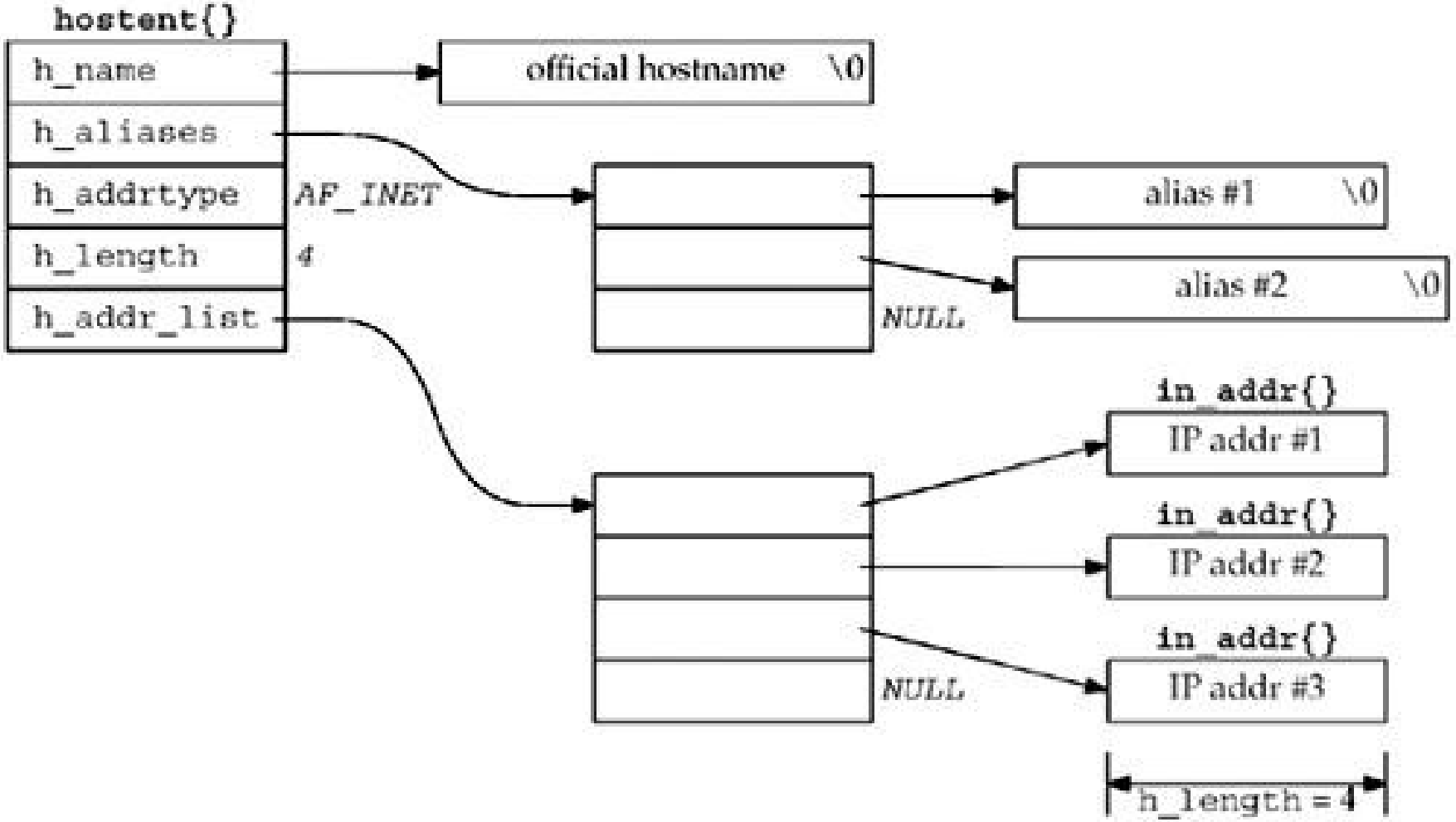
- A maioria das aplicacoes usam nomes e não enderecos IP.

gethostbyname

```
#include <netdb.h>

struct hostent *gethostbyname (const char *hostname);

Returns: non-null pointer if OK,NULL on error with h_errno set
```

```
struct hostent {
  char *h_name;      /* official (canonical) name of host */
  char **h_aliases; /* pointer to array of pointers to alias names */
  int h_addrtype;   /* host address type: AF_INET */
  int h_length;     /* length of address: 4 */
  char **h_addr_list; /* ptr to array of ptrs with IPv4 addrs */
};

#define h_addr    h_addr_list[0] /* for backward compability */
```

Diagrama da estrutura *hostent*



gethostbyname

- Erro é retornado em **`h_errno`** (**`netb.h`**)
  - HOST_NOT_FOUND
  - TRY_AGAIN
  - NO_RECOVERY
  - NO_DATA

- Função `hstrerror` detalha o erro

Exemplos de saídas

```
solaris % hostent gemini.tuc.noao.edu
official hostname: gemini.tuc.noao.edu
         address: 140.252.1.11
         address: 140.252.3.54
         address: 140.252.4.54
         address: 140.252.8.54
```

Exemplos de saídas

```
solaris % hostent www
official hostname: bsdi.kohala.com
          alias: www.kohala.com
          address: 206.62.226.35
```

Resolver para endereços IPv6

- Opção para retornar somente endereço Ipv6
  - Na aplicação:

```
res_init();
res.options |= RES_USE_INET6;
```

  - Por usuário

```
export RES_OPTIONS=inet6
```

  - No sistema

```
/etc/resolv.conf
options inet6
```

gethostbyaddr

```
#include <netdb.h>
struct hostent *gethostbyaddr (const char *addr,
                               socklen_t len, int family);

Returns: non-null pointer if OK, NULL on error with h_errno set
```

***addr** é ponteiro para **in_addr** ou **in_addr6** contendo endereço IPv4 ou IPv6

- **len** - tamanho da estrutura 4 (IPv4) ou 16 (IPv6)
- **family** – AF_INET ou AF_INET6

uname

- Retorna nome da estação (mesma idéia do comando dos SOs Unix-like)

```
#include <sys/utsname.h>

int uname(struct utsname *name);

Returns: nonnegative value if OK, -1 on error
```

```
#define _UTS_NAMESIZE   16
#define _UTS_NODESIZE   256

struct utsname {
  char  sysname[_UTS_NAMESIZE];   /* name of this operating system */
  char  nodename[_UTS_NODESIZE]; /* name of this node */
  char  release[_UTS_NAMESIZE];   /* O.S. release level */
  char  version[_UTS_NAMESIZE];   /* O.S. version level */
  char  machine[_UTS_NAMESIZE];   /* hardware type */
};
```

gethostname

- Retorna nome da estação (semelhante ao `uname` mas mais simples)

```
#include <unistd.h>

int gethostname( char *name, size_t namelen);

Returns: 0 if OK, -1 on error
```

getservbyname

- Retorna nome do serviço (Lê o `/etc/services`)

```
#include <netdb.h>

Struct servent * getservbyname( const char *servname,
                                const char * protoname);

Returns: nonnull pointer if OK, NULL on error
```

```
struct servent {
  char   *s_name;      /* official service name */
  char  **s_aliases;   /* alias list */
  int     s_port;      /* port number, network-byte order */
  char   *s_proto;     /* protocol to use */
};
```

Arquivo /etc/services

```
castudillo@quechua:~$ cat /etc/services
tcpmux    1/tcp             # TCP port service multiplexer
daytime       13/tcp
daytime       13/udp
ftp-data  20/tcp
ftp    21/tcp
fsp    21/udp    fspd
ssh    22/tcp            # SSH Remote Login Protocol
ssh    22/udp
telnet    23/tcp
smtp      25/tcp    mail
time      37/tcp    timserver
time      37/udp    timserver
tacacs    49/udp
re-mail-ck    50/tcp           # Remote Mail Checking Protocol
re-mail-ck    50/udp
domain    53/tcp           # Domain Name Server
domain    53/udp
```

getservbyport

- Procura serviço pela porta + protocolo

```
#include <netdb.h>

Struct servent * getservbyport(int port, const char * protoname);

Returns: nonnull pointer if OK, NULL on error
```