Una prima applicazione in C per l'utilizzo delle socket

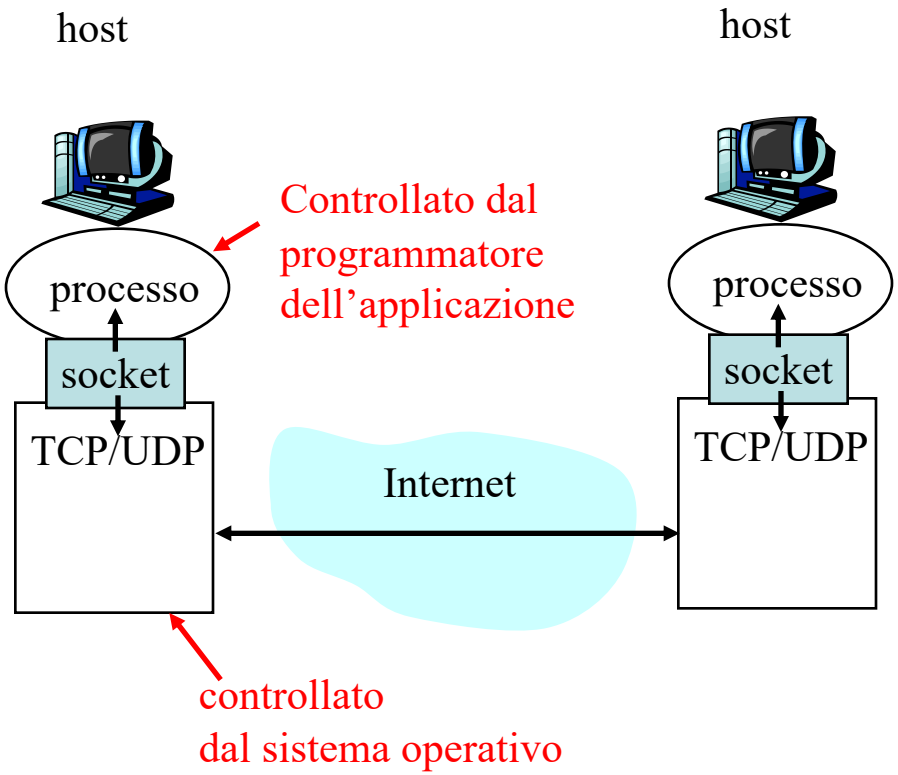
Richiamo sulle socket

Socket in Windows (WinSock)

Differenze tra sistemi operativi

Socket API

- API: Application Programming Interface
- Una socket e' un dispositivo di interfaccia tra un processo e il sistema operativo
- Un processo può sia spedire messaggi a un altro processo che ricevere messaggi mediante la propria socket
- La comunicazione via socket usa il modello di I/O di Unix
  - open-read-write-close
  - file descriptor = (pathname, flag)



Scelta del tipo di servizio

TCP - orientato alla connessione

- Il client stabilisce la connessione al server
- Il client e il server si scambiano messaggi multipli di dimensione arbitraria
- Il client termina la connessione

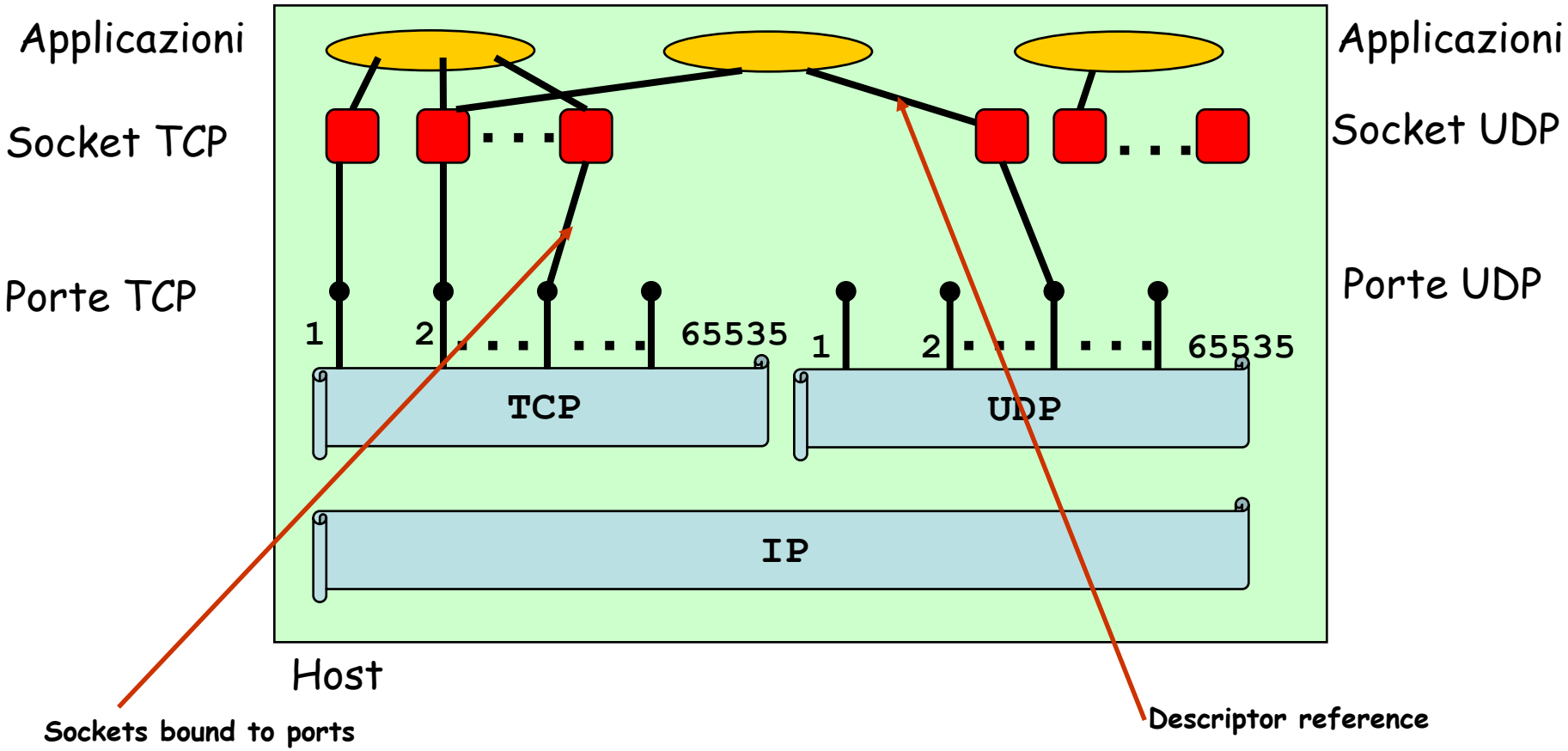
Garantisce l'affidabilità del servizio

UDP - privo di connessione

- Il client costruisce il messaggio
- Il client spedisce il messaggio al server
- Il messaggio deve entrare in un datagramma UDP (65.500 bytes)
- Il server risponde

Il servizio non è affidabile

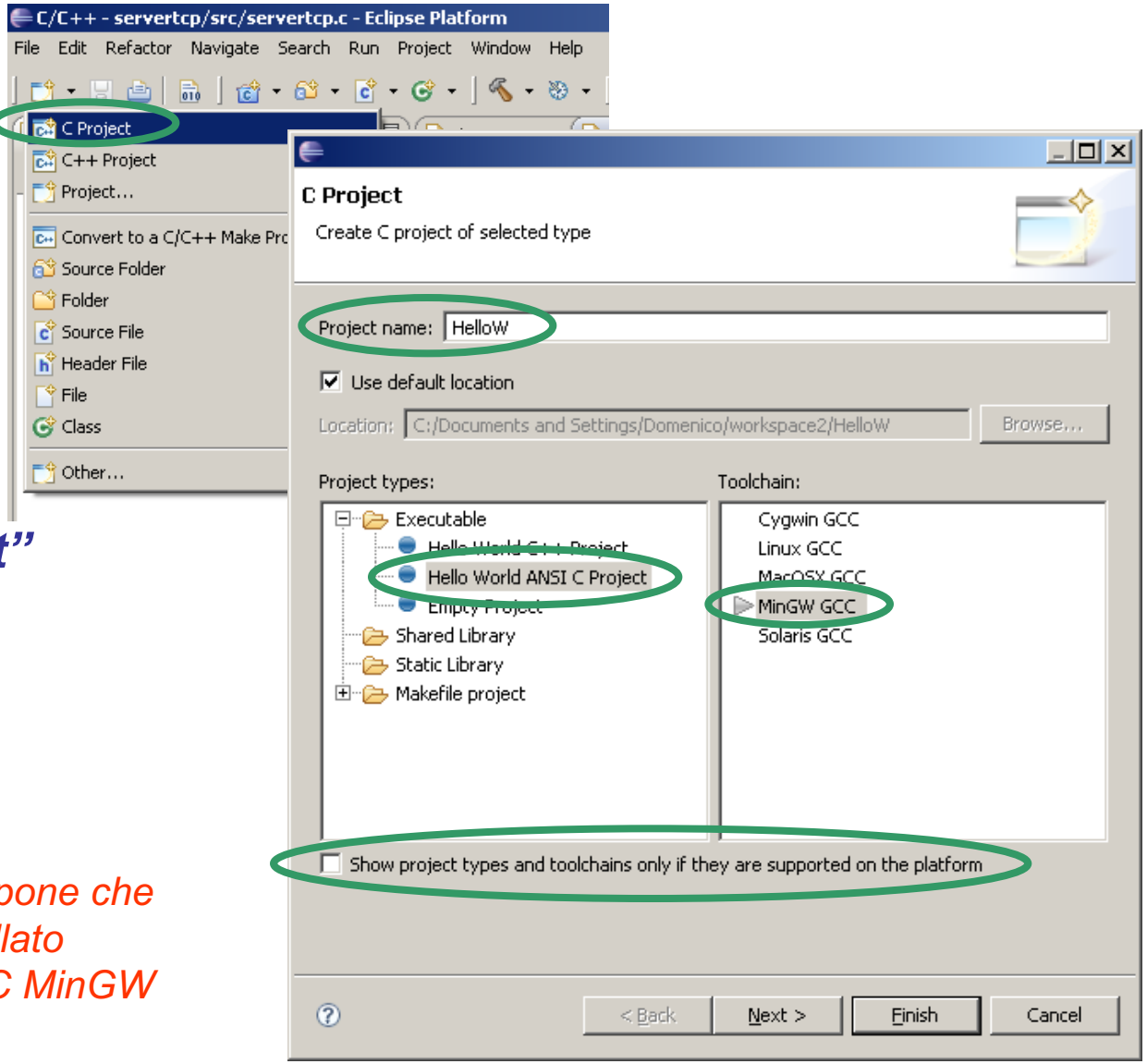
Socket e numeri di porta



Creare un Progetto con Eclipse CDT

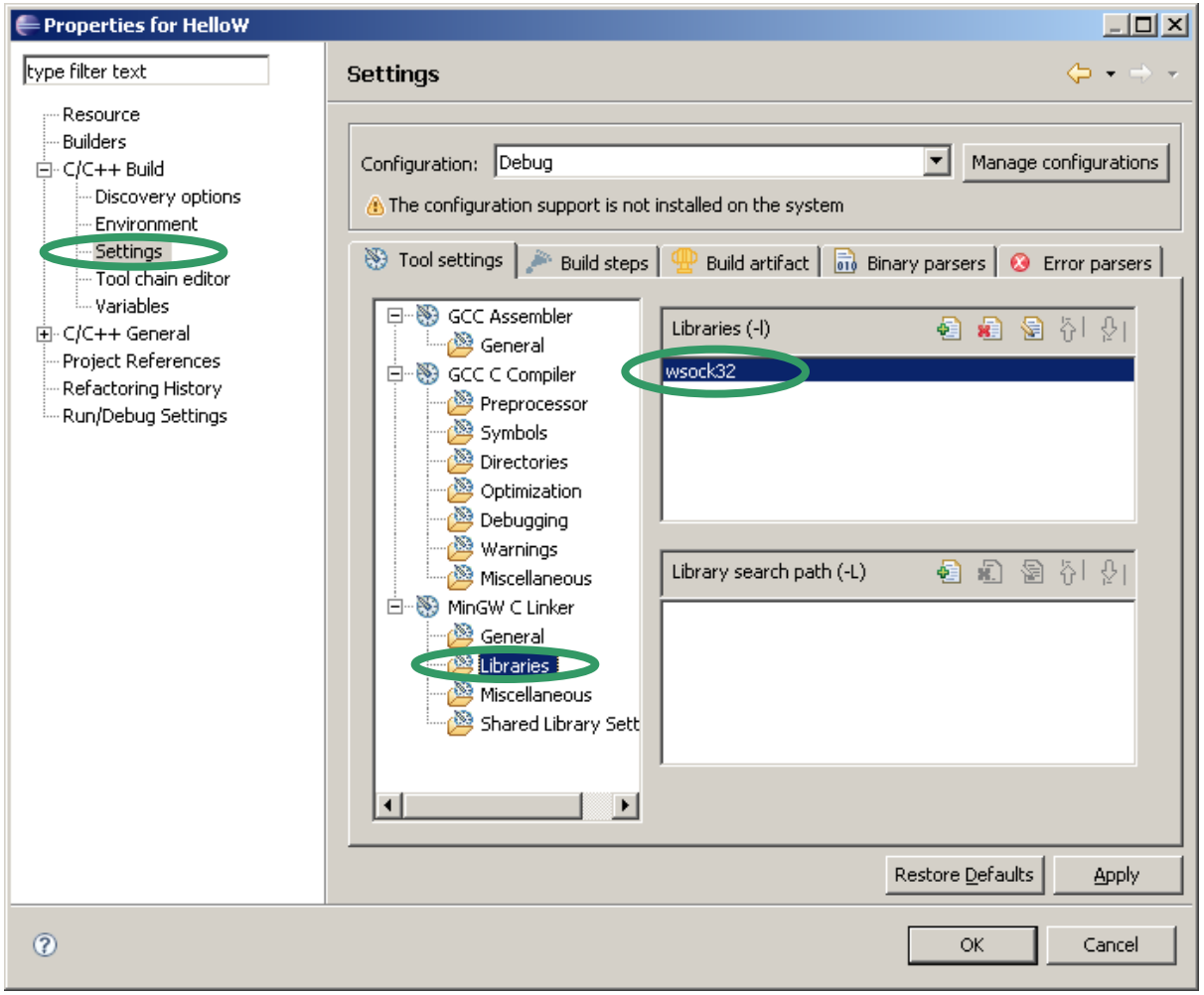
- Menù “*File*” → “***New***”
- Sel. “***C Project***”

- *Project name: **“…”***
- *Deselezionare **“Show project types…”***
- *Selezionare **“Hello World ANSI C Project”***
- *Selezionare nella toolchain **“MinGW GCC”***



NB: Questo esempio presuppone che sulla macchina sia installato correttamente il compilatore C MinGW

Aggiungere la libreria Winsock al progetto (in Eclipse CDT)



- Project Properties-> ***C/C++ Build***
- Settings ->
- MinGW C Linker ***-> Libraries***
- Basta inserire il nome della libreria (***wsock32*** o ***ws2_32*** per *Winsock2*) senza il prefisso ***lib*** e l'estensione ***.a***

Windows: inizializzazione dell'applicazione

- Tutte le applicazioni Winsock devono essere inizializzate per essere sicuri che le socket windows siano supportate dal sistema
- Per inizializzare Winsock:
  - Creare un elemento di tipo WSADATA:

```
WSADATA wsaData;
```

…inizializzazione dell'applicazione

- La struttura WSADATA contiene informazioni sull'implementazione delle socket windows

```
typedef struct WSAData {
  WORD wVersion;
  WORD wHighVersion;
  char szDescription[WSADESCRIPTION_LEN+1];
  char szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short iMaxSockets;
  unsigned short iMaxUdpDg;
  char FAR* lpVendorInfo;
} WSADATA
```

wVersion: versione per la specifica di socket utilizzata

wHighVersion: versione massima supportabile per la specifica di socket windows

…inizializzazione dell'applicazione

- Dopo aver creato un elemento di tipo WSADATA

```
WSADATA wsaData;
```

- Specificare la versione di socket windows richiesta e recuperare i dettagli dell'implementazione della socket windows specifica

```
Int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);
```

- Restituisce 0 in caso di successo. Il codice di errore altrimenti

```
If (iResult != 0)
        printf("error at WSASturtup\n");
```

…inizializzazione dell'applicazione

```
Int iResult = WSAStartup(MAKEWORD(2 ,2), &wsaData);

If (iResult != NO_ERROR)
        printf("error at WSASturtup\n");
```

specifica il numero di versione di winsock sul sistema e lo costruisce correttamente

wVersionRequested: versione di socket windows che il chiamante può usare.

```
int WSAStartup(
  WORD wVersionRequested,
  LPWSADATA lpWSAData
);
```

lpWSAData : puntatore alla struttura WSDATA che contiene informazioni per l'implementazione della socket

Un esempio di codice

```
#include <stdio.h>
#include <winsock.h>
int main() {
// Initialize Winsock
WSADATA wsaData;
WORD wVersionRequested;
wVersionRequested= MAKEWORD(2,2);
int iResult = WSAStartup(wVersionRequested, &wsaData);
   if (iResult != NO_ERROR) {
        printf("Error at WSAStartup()\n");
        printf("A usable WinSock DLL cannot be found");
        return -1;
   }

// The WinSock lib is accessible. Proceed
printf("No errors occurred. \n");
system("pause");
return 0;
}
// main end
```

Differenze tra sistemi operativi

- Unix/Mac

```
//Header files
#include <sys/socket.h>
//per socket(), bind()  connect()
#include <unistd.h>
// per close()
#include <arpa/inet.h>
// per sockaddrin
...
// No inizialization
   required
...


// Shutdown
close(Mysocket);
```

- Windows

```
//Header files
#include <winsock.h>
...
// Initialize Winsock
 WSADATA wsaData;
 int iResult =
   WSAStartup(MAKEWORD(2,2),&wsaData);
 if (iResult != 0) {
  printf("Error at WSAStartup()\n");
  return 0;
 }
...
// Shutdown
closesocket(Mysocket);
WSAcleanup();
```

Scrivere un programma portabile

```
#if defined WIN32
#include <winsock.h>
#else
#define closesocket close
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#endif

#include <stdio.h>
...
```

```
int main() {

...
   #if defined WIN32
   // Initialize Winsock
   WSADATA wsaData;
   int iResult =
        WSAStartup(MAKEWORD(2,2),
   &wsaData);
   if (iResult != 0) {
        printf("Error at WSAStartup()\n");
        return 0;
   }
   #endif
   int Mysocket;

...
   closesocket(Mysocket);
   #if defined WIN32
          WSACleanup();
   #endif
   return 0;
} // main end
```