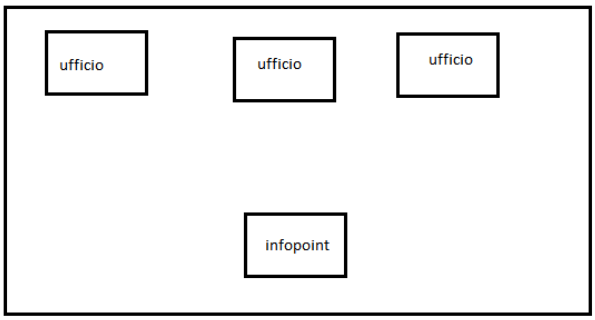
Parte 1- Organizzazione della rete

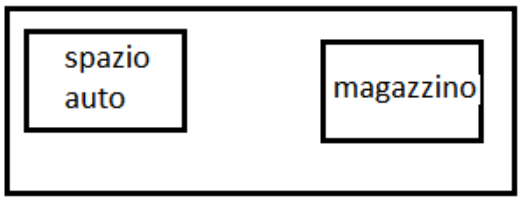
Secondo le indicazioni della traccia, esistono due tipologie di officine

Officina principale



officina principale

Officina periferica



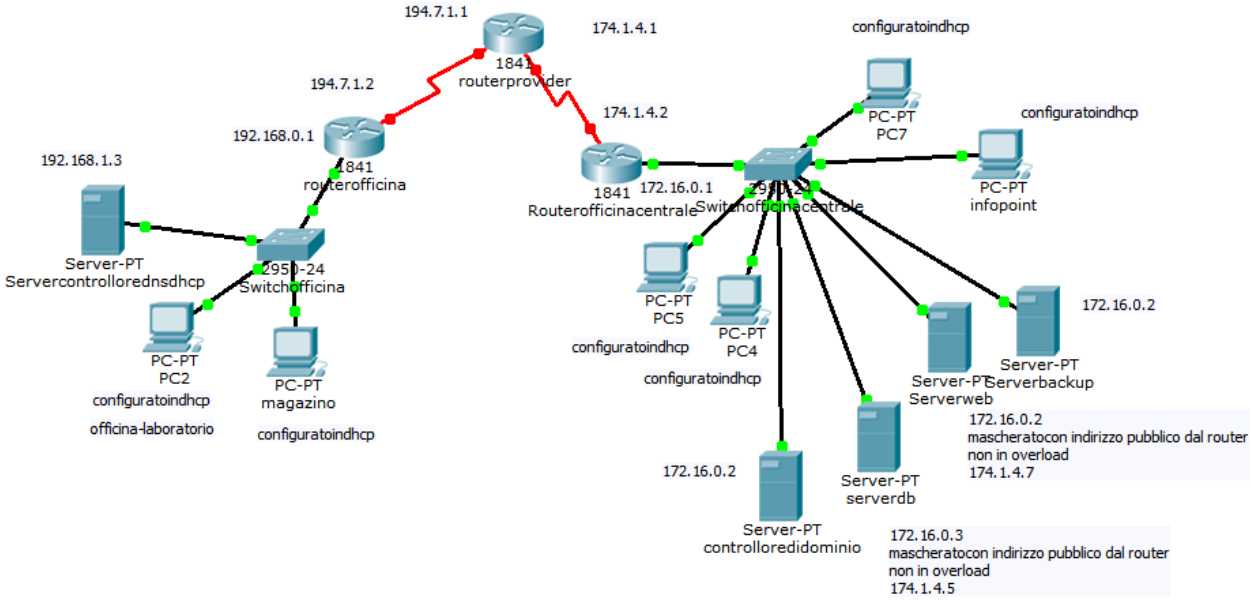
officina periferica

Entrambe dotate di connessione Internet, ovviamente tramite provider

Possiamo pensare ad una struttura di rete client server per entrambe le realtà. Nell'officina principale la scelta è obbligata.

Nello schema che segue rappresentiamo la struttura di rete per il collegamento di una sede periferica generica alla sede centrale.

Si riportano gli indirizzi privati e pubblici, considerando che i server con visibilità esterna devono avere un mascheramento con indirizzo pubblico fisso.



Passiamo al cablaggio.

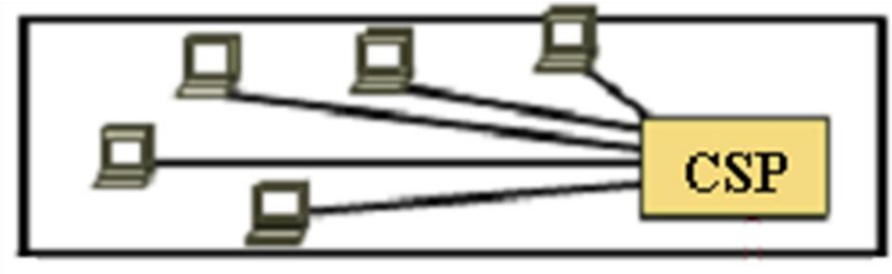
Considerata l'indicazione della traccia, il tipo di rete deve essere 802, quindi pensiamo ad ethernet.

Per questioni di semplicità non consideriamo la possibilità, certamente plausibile, di una connessione wifi.

Consideriamo anche che le due sedi abbiano una dislocazione su un solo piano.

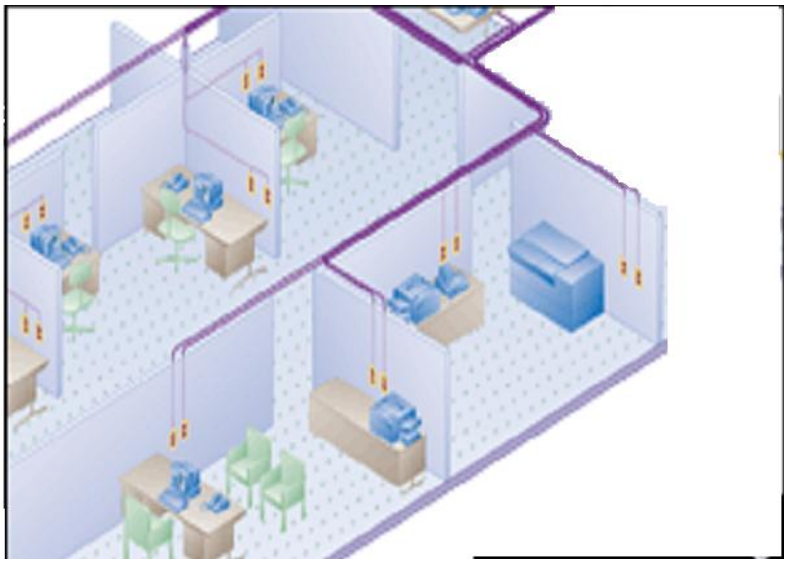
Tenendo conto delle indicazioni dello standard IEEE per il cablaggio , possiamo riferirci ai seguenti schemi di riferimento.

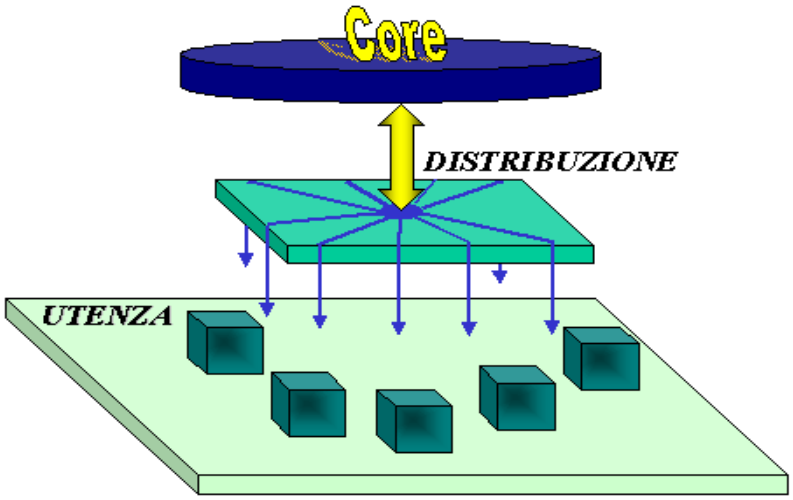
PIANO

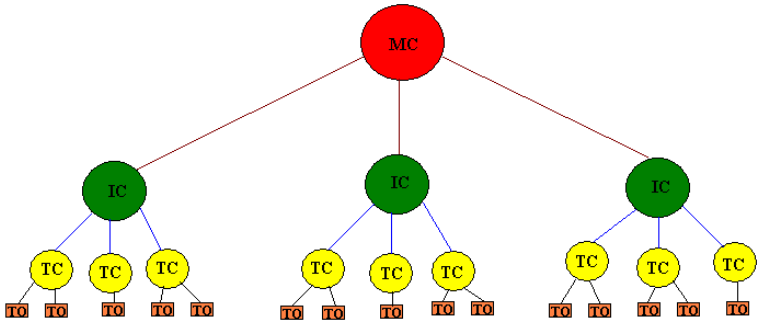


CS=Centro stella

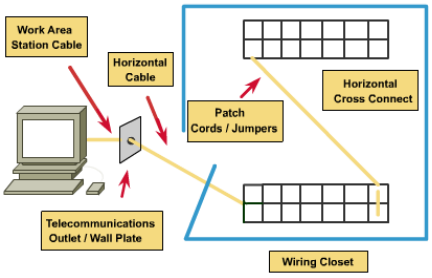
CSP=Piano

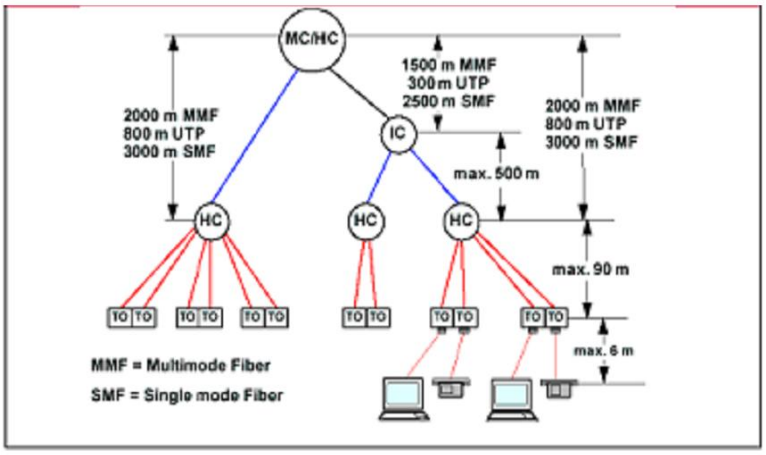



Core
DISTRIBUZIONE
UTENZA


MC
IC
IC
IC
TC
TC
TC
TC
TC
TC
TC
TC
TC
TO

Horizontal Cabling Component





PARTE 2- Seguito cablaggio e Programmazione di rete

Considerazioni sulle scelte di cablaggio.

Lo standard di riferimento è EIA.

Una qualsiasi rete deve essere ripartita in tre zone: **core, distribution, work area**, come rappresentato in una figura precedente (parte1 della soluzione).

La struttura dei collegamenti deve essere per dorsali. Trattandosi di collocazione su singolo piano vale una connessione col centro stella di piano, indicato con CSP nella figura riportata nella parte1 della soluzione.

Ogni computer deve, di fatto, essere collegato ad una dorsale.

Per l'officina periferica possiamo pensare che un collegamento a 100Mb con cavo di categoria 5 vada bene.

Lo switch, abbinato ad un pathc panel, deve essere collocato in un wiring closet o in un armadio. Senza soffermarci troppo sui vincoli che la normativa pone per la qualificazione e l'ubicazione di questi ambienti, ricordiamo che il collegamento del cavo proveniente dal computer singolo deve raggiungere prima il patch panel e poi, con un cavo che può variare da 1,5 a 5m, deve raggiungere lo switch. La distanza massima di collegamento del computer deve essere 100m. Considerato che, come vedremo più avanti, anche nell'area di lavoro ci deve essere un cavo patch, con le stesse caratteristiche di quello per il wiring closet, ne deriva che la dorsale di piano deve essere 90m al massimo.

I cavi. Abbiamo detto categoria 5. Considerato che l'ambiente interessato è esposto a disturbi, non conviene usare cavi UTP, ma almeno STP o FTP.

In entrambi i casi vanno rispettate le regole di sicurezza sintetizzabili nella sigla LSFR0H, che impone cavi non emettano fumi, ritardino la fiamma e non emettano alogeni.

Il passaggio dei cavi deve avvenire o sotto il pavimento o nel controsoffito. Considerando che si tratta di officine, quindi di capannoni, è facile la soluzione del controsoffitto. Una delle figure della parte1 riporta le dorsali che passano appunto sopra le pareti divisorie.

Per la work area dobbiamo considerare che ogni postazione abbia almeno due connetttori RJ45 per la rete, due prese unel bipasso da 3 A, due prese schuko.
Il collegamento tra la postazione la TO (terminazione utente, secondo la nomenclatura EIA) deve essere realizzata con un cavo da 1,5-5m.
Ultima considerazione per il cablaggio dell'officina principale. Considerata la presenza di server con consistente carico di lavoro, conviene pensare ad un **sezionamento asimmetrico**, prevendo sullo switch porte da 1 GB, almeno per i server web e di database.

Prima di chiudere questa parte, un piccolo cenno alla programmazione dei router utilizzando lo specifico linguaggio CISCO.

Ci riferiamo al router dell'officina periferica .

```
Router>enable
Router#config t

Router(config)#interface fa0/0
Router(config-if)#ip address 192.168.0.1 255.255.255.0
Router(config-if)#no shutdown
Router(config)#interface se2/0
Router(config-if)#ip address 194.7.2.2  255.255.255.0
Router(config-if)#clock rate 64000
Router(config)#router rip
Router(config-router)#network 194.7.1.0
Router(config-router)#network 192.168.0.0
```

Passiamo alle altre parti della traccia

- Sistemi operativi per le reti
- Software per lo scambio di dati tra sedi periferiche e centrale
- Software per la generazione della pagina web e gestione degli accesssi alle funzioni prevviste
- Software per la gestione del DB

Non ci soffermiamo, per questioni di tempo e di spazio sul primo punto al quale riserviamo poche considerazioni. Il sistema operativo dovrebbe essere windowserver come controllore di dominio, server dns locale, server dhcp e atcive directory.

Per la parte di database e serverweb potremmo optare per apache e linux.

Il secondo punto, espresso nella traccia come "architettura protocollare" può essere pensato come un applicativo che si avvalga dei socket per la connessione.

Ipotizzando di usare l'opzione TCP prevista dai socket non è necessario pensare ad un sw che faccia anche il controllo di errori. Certamente però è preferibile una connessione che usi una fase preliminare di handshake, mutuabile da quella a tre vie del TCP.

Dovremmo quindi avere tre fasi:

- Connessione con handshake
- Scambio dati
- Chiusura del canale

Per lo scabio dei dati, considerato che i socket lavorano su messaggi, possiamo pensare di strutturare i dati in struct e poi integrarle nel messaggio effettivamente spedito.

Riportiamo di seguito il software per lo scambio di messaggi tra un Client ed un Server utilizzabile come modello di riferimento generale.

SERVER

```
#include <windows.h>
#include <winsock.h>
#include <stdio.h>
#include <conio.h>
#define MAX_STRING 256
int main(){
	SOCKET serversock,client;
	struct sockaddr_in serversock_addr,client_addr;
	WSADATA data;
	WORD p;
	int len(char *);
	int lun, err=0;
	char mex[]="CONNESSIONE AVVENUTA CON SUCCESSO!\n";

	p=MAKEWORD(2,0);
	err=WSAStartup(p,&data);
	serversock=socket(PF_INET,SOCK_STREAM,0);
	serversock_addr.sin_family=PF_INET;
	serversock_addr.sin_port=htons(1550);
	serversock_addr.sin_addr.s_addr=INADDR_ANY;
	err=bind(serversock,(struct sockaddr*)&serversock_addr,sizeof(struct sockaddr_in));
	err=listen(serversock,1);
	lun=sizeof (struct sockaddr);
	client=accept(serversock,(struct sockaddr*)&client_addr,&lun);
	if (err!=0) printf ("Errore connessione");
	send(client,mex,40,0);
	printf ("connessione avvenuta");
	closesocket (client);
	closesocket(serversock);
	WSACleanup();
		return 0;

}
```

CLIENT

```
#include <windows.h>
#include <winsock.h> //includo le varie librerie tra cui quella winsock
#include <stdio.h>
#include <conio.h>

int main()
{
	int n=0, i=0;
	char mex[100];
```

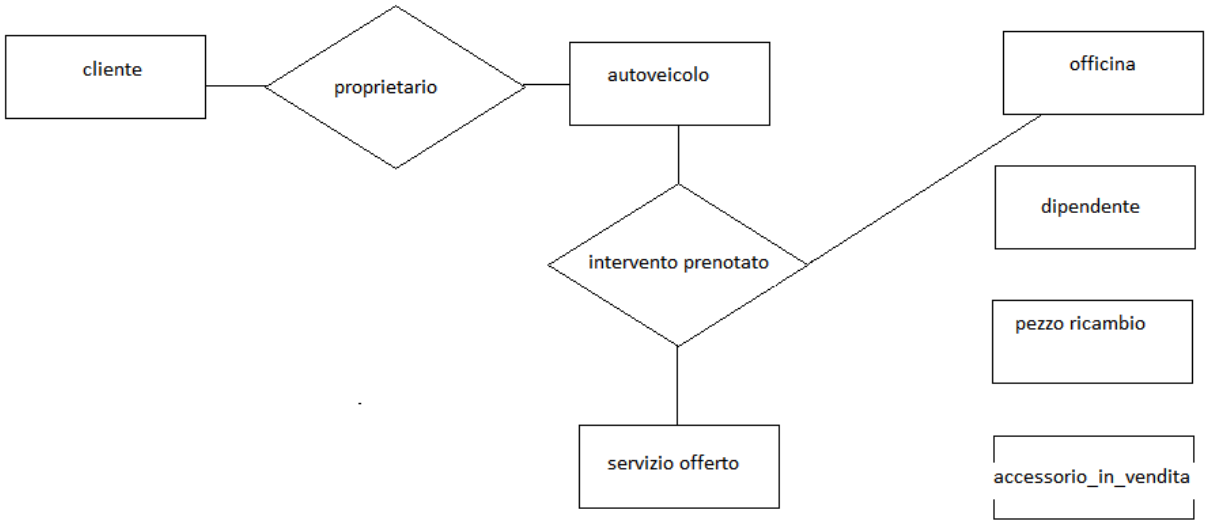
```
SOCKET sock;
struct sockaddr_in sock_addr;
WSADATA data;
WORD p;
p=MAKEWORD(2,0);
WSAStartup(p,&data);
sock=socket(AF_INET,SOCK_STREAM,0);
sock_addr.sin_family=AF_INET;
sock_addr.sin_port=htons(1550);
sock_addr.sin_addr.s_addr =inet_addr( "174.1.4.7" );
connect(sock,(struct sockaddr*)&sock_addr,sizeof(struct sockaddr));
while(n==0)
    n=recv(sock,mex,100,0);
//prevedo una stampa di controllo, da eliminare dopo il debug
printf("%d'\n'",n);
while(mex[i]!='\0')
{printf("%c \n",mex[i]);
i++;
}
closesocket(sock);
WSACleanup();
return 0;
}
```

Parte Web e DB

Si tratta di un problema classico di progettazione e implementazione di un database che richiede un'analisi puntuale delle funzionalità di sistema nel suo complesso, in modo da ottimizzare tempi di accesso e impegno di memoria.

Risulta fondamentale lo schema concettuale e logico del database che evita la duplicazione dei dati e, insieme ad un sistema di backup automatico, consente, in caso di crash, il recupero dei dati.

Riportiamo di seguito il modello ER:



Il modello logico si articola intorno alle seguenti relazioni fondamentali:

officina(codice, denominazione, indirizzo)
servizio_offerto(codice, descrizione, costo)
accessorio_in_vendita(codice, descrizione, costo_unitario, quantità)
pezzo_ricambio(codice, pezzo, descrizione, costo_unitario, quantità)
cliente(codice, nome, cognome, telefono, indirizzo)
autoveicolo(codice, cliente, numero_telaio, anno_costruzione)
intervento_prenotato(codice, servizio_offerto, autoveicolo, officina, data_prenotazione)
dipendente(codice, nome, cognome, telefono, indirizzo, login, password)

Il database in questione può essere implementato con il DBMS MySQL, un software open source piuttosto semplice, multipiattaforma, adatto ad accessi da web in quanto dotato di un modulo di integrazione con linguaggi di scripting. Il database deve essere disponibile su un server on line gestito da un ambiente tipo IIS o Apache. Strumenti specifici per la grafica, tipo DreamWeaver ed un lingiaggio di programmazione tipo PHP, che consente di arricchire le pagine web con script eseguibili direttamente sul server, sono una risorsa potente per lo sviluppo dell'applicazione richiesta.
In particolare, PHP consente di implementare un motore di scripting server side molto diffuso e multipiattaforma con un buon supporto alla connettività, attraverso componenti standard a database in generale e a MySQL in particolare.
Ovviamente, per potere utilizzare Php è necessario aver installato sul proprio sistema un Web Server.

Una possibile home page del sito della casa automobilistica è la seguente; in essa ogni icona o descrizione è cliccabile e permette di scaricare il relativo documento pdf.

<table>
<tr><td colspan="3">Intestazione: logo e nome dell'officina centrale</td></tr>
<tr><td colspan="3">Prodotti >> Cataloghi da scaricare >> Tutto per l'officina</td></tr>
<tr><td>Rete delle officine

Servizi offerti

Prodotti
Autovetture e veicoli commerciali leggeri
Motociclette e motorini
Accessori
Cataloghi da scaricare</td><td>Immagini di prodotti e servizi offerti con didascalia</td><td>Area riservata ai dipendenti

Nome

Password

Login</td></tr>
<tr><td colspan="3">Indirizzo dell'officina principale</td></tr>
</table>

L'area riservata ai dipendenti viene realizzata tramite un form HTML con due campi di input e un pulsante di invio:

```
<html>
	<body>
		<form action="autenticazione.php" method="post">
			Nome: <input type="text" name="user"><br>
			Password: <input type="text" name="pw"><br>
		<input type="submit" name="submit" value="Login">
		</form>
	</body>
</html>
```

Quando il dipendente compila il modulo sopra e fa clic sul pulsante di invio, i dati del form vengono inviati per l'elaborazione al file PHP denominato "autenticazione.php". I dati del modulo vengono inviato con il metodo HTTP POST.

La casella di testo di nome passa il suo valore alla pagina autenticazione.php che recupera il contenuto della variabile utilizzando $user e $pw:

```
<?php

	// define variables and set to empty values

	$user = $pw =  " ";
	if ($_SERVER["REQUEST_METHOD"] == "POST")

	{
		$name = test_input($_POST["user"]);
		$pwd = test_input($_POST["pw"]);
	}
}

?>
```

I valori delle variabili user e pw vengono utilizzati di seguito nello script di connessione che si collega al server contenente il database dell'officina, ovvero i cataloghi dei servizi offerti, degli accessori e dei pezzi di ricambio.

nel caso particolare possiamo ipotizzare una specificazione del tipo:

```
$ hostname= “174.1.4.7”;

$ database = “catalogo_accessori”;

$ mysqli = new mysqli( $hostname,$user, $pw,”database);
```

ne deriva il codice php che segue

```
<?php
$ hostname= “174.1.4.7”;
$ database = “catalogo_accessori”;
$ mysqli = new mysqli( $hostname,$user, $pw,”database”);
// verifica dell'avvenuta connessione
if (mysqli_connect_errno()) {
          echo "Errore : ".mysqli_connect_error();
          // interruzione delle esecuzioni i caso di errore
       exit();
 }
else {
          // notifica in caso di connessione attiva
       echo "Connessione avvenuta con successo";
}
 // chiusura della connessione
$mysqli->close();
?>
```