PROTOCOLLI E TELECOMUNICAZIONI
ALLA RICERCA DEL VALORE AGGIUNTO

Tracciare l'evoluzione dei protocolli significa tracciare l'evoluzione delle telecomunicazioni. Ovvero parlare di ICT anziché di TLC. Il valore aggiunto, insito nei protocolli di rete, si è evoluto passando dalla correzione d'errore, alla gestione del dialogo, alle funzioni collaborative per arrivare infine ai servizi applicativi, sempre più sofisticati e orientati all'erogazione dei servizi web. La pietra miliare è il modello OSI, sempre attuale, rivisto alla luce delle nuove tecnologie di rete wireless e dalla necessità di definire protocolli che supportino ambienti multimediali.

Andrea Baiocchi
Giacomo Zanotti

3·4

1. INTRODUZIONE

L'enciclopedia gratuita offerta su Internet, la ormai famosa Wikipedia (http://en.wikipedia.org/wiki/Main_Page), fornisce la definizione seguente del termine protocollo di comunicazione: *a set of rules governing communication between electronic devices*.
Dobbiamo notare che il generico termine protocollo è riferito a varie altre categorie, oltre alle comunicazioni: diplomazia e politica, informatica, medicina e scienza, letteratura e spettacoli. È interessante il fatto che per cercare la definizione corrente del termine protocollo con le sue varie accezioni, inclusa quella qui oggetto di trattazione nell'ambito delle telecomunicazioni, sono state invocate una complessa messe di protocolli di comunicazione: l'HTTP per dialogare con il programma applicativo che gestisce le pagine web di Wikipedia; i protocolli della famiglia TCP/IP per effettuare un trasferimento affidabile dei dati dal server remoto alla macchina client, per risalire dal nome del destinatario al suo indirizzo di rete, per determinare un percorso interno alle reti attraversate fino all'indirizzo di destinazione, per mantenere la continuità del collegamento; i protocolli specificati negli standard della famiglia IEEE 802 per accedere attraverso la rete locale Ethernet alla quale la macchina cliente è connessa nell'esempio in esame. Ma andiamo con ordine, per dipanare la matassa di sigle che si affastella non appena si esamina anche la più semplice attività di interconnessione in rete.
Il problema base della comunicazione tra entità remote può essere schematizzato come segue: una popolazione di entità, consistenti in esseri umani o processi eseguiti automaticamente da macchine, in particolare computer, e distribuiti geograficamente su una data area, hanno necessità di cooperare con altre entità facenti parte della medesima popolazione per conseguire i propri obbiettivi funzionali. La "cooperazione" può essere vista in generale come l'evoluzione di una macchina a stati distribuita, in cui ogni entità svolge azioni che dipendono dai dati e programmi/modi di procedere residenti localmente e dalle sollecitazioni esterne, in particolare dai risultati delle comunicazioni con altre entità della po-

polazione. Informalmente, i protocolli specificano le regole di questo tipo di interazione.
Un semplice esempio, per rendere meno astratta la definizione, ma che non rende necessariamente l'intera complessità che vi si nasconde, è una popolazione di esseri umani dispersi in un dato territorio che necessitano di effettuare conversazioni a voce. Il problema di comunicazione consiste, ridotto all'osso, nel fornire mezzi per cercare e allertare un destinatario da parte di un originatore, nel permettere il trasferimento intelleggibile della voce in modo bidirezionale tra le due parti durante la conversazione, eventualmente impiegando risorse di comunicazione condivise dai membri della popolazione di utenti, nel fornire i mezzi per terminare ordinatamente la conversazione e rilasciare eventuali risorse condivise utilizzate. Altro esempio è l'esecuzione di transazioni tra macchine (*mail server*) per la distribuzione della posta elettronica. Anche in questo caso il paradigma comporta la necessità di scoprire i potenziali partner, contattarli, negoziare le condizioni per l'esecuzione della transazione, trasferire dati e informazioni di controllo relativi alla transazione tra le macchine coinvolte, quindi terminare la transazione. Ogni passo di questo processo in ogni mail server è controllato da un programma (software). Questo software elabora i contenuti informativi scambiati (per esempio esegue traduzioni di codifica, memorizza in archivi) ed esegue i protocolli di comunicazione che permettono la cooperazione tra i processi residenti su macchine remote. In ambedue gli esempi, conversazioni e distribuzione della posta, sistemi intermedi con i loro protocolli, programmi e dati intervengono ad un livello più "basso" di quello dell'applicazione. Scopo dei sistemi intermedi è fornire e controllare la connettività e la qualità del servizio di rete: in sintesi, trasferire dati all'interno della popolazione di utenti in modo tecnicamente efficiente (date le tecnologie disponibili) ed economicamente conveniente (dati gli obiettivi e il valore dei servizi supportati).
Cos'è quindi un protocollo di comunicazione? La definizione data all'inizio lo identifica come un insieme di regole che permettono uno scambio di informazioni tra apparati elettronici, o più in generale, come illustrato nei due semplici esempi, tra processi applicativi (in senso lato, inclusi dunque gli esseri umani). È cruciale precisare il termine "scambio": lo scambio di informazioni fa pensare a un passaggio di dati da un'entità ad un'altra, eventualmente in modo bidirezionale. Focalizzare l'attenzione sulla fase di "scambio" dei dati elude però la parte forse più importante di un protocollo che è il "controllo" della scambio, cioè la definizione di procedure distribuite che permettono alle entità coinvolte di arrivare a scambiare i dati, controllando le condizioni tecnico-economiche nelle quali lo scambio avviene (ecco il valore aggiunto!).
A questo scopo un generico protocollo comprende tre parti costitutive:

- una *sintassi*, che precisa le strutture dei messaggi (formato, parti componenti, possibili valori attribuibili alle varie parti) usati nel protocollo con finalità di controllo e di trasferimento di dati;
- una *semantica*, che stabilisce il significato dei particolari valori inseriti nelle parti di ogni messaggio e degli eventi che si presentano nell'evoluzione del protocollo, anche in rapporto alle sequenza di messaggi scambiati a partire dall'inizio dell'interazione, e definisce quindi le azioni che una entità deve intraprendere; si tratta in sostanza della specificazione della logica della macchina a stati eseguita da ogni entità partecipe del protocollo;
- *temporizzatori*, che presiedono alla corretta evoluzione nel tempo del protocollo, ovvero delle macchine a stati associate al protocollo in ogni entità che prende parte alla sua esecuzione; scopo essenziale dei temporizzatori (*timer*) è regolare la "marcia" dell'evoluzione del protocollo, verificando che le transizioni di stato avvengano in lassi di tempo coerenti con una corretta funzionalità.

Nel riquadro di p. 23 è illustrato qualche esempio per ognuno dei tre concetti.
Notiamo che, in un protocollo, le funzioni "collaborative", anche quando comportano attività basilari, possono richiedere l'utilizzo di algoritmi molto complessi o anche condurre a situazioni (stati dell'automa) "indecidibili". In questo ultimo caso, il protocollo non è in grado di risolvere univocamente la situazione ma si limita ad "avvisare" tutti gli attori che partecipano al processo collaborativo che si è verificata una situazione anomala e che dunque è necessa-

Esempi di aspetti sintattici, semantici e di temporizzazione dei protocolli

Un'unità d'informazione (pacchetto) di un protocollo per il trasferimento dei dati in rete, quale per esempio IP, è descritta da una sequenza di ottetti (byte) raggruppati a formare cosiddetti campi. Ogni campo può avere solo determinati valori (dominio) e rappresenta un elemento d'informazione portato dal pacchetto: per esempio alcuni byte contengono l'indirizzo del mittente del pacchetto (interfaccia di rete dalla quale è originato il pacchetto), altri, l'indirizzo del destinatario (interfaccia alla quale è connessa l'entità destinataria finale del pacchetto). Altri byte possono contenere codici per il controllo di errori, altri il numero della versione del protocollo (anche i protocolli di comunicazione hanno una vita ed evolvono, come il software che installiamo sui nostri computer: hanno quindi un numero di versione, per motivi di verifica di compatibilità).

Seguendo il filo di questo esempio, la semantica specifica la sequenza di operazioni da fare con un pacchetto da parte di una generica entità di rete che lo riceve. Per esempio:

1. verifica la presenza di errori nel pacchetto;
2. se corretto, verifica se trattasi di versione compatibile con quelle disponibili localmente;
3. verifica se l'indirizzo destinatario coincide con uno degli indirizzi attribuiti all'entità;
4. in caso contrario, verifica se rientra tra gli indirizzi di una porzione di rete connessa all'entità, oppure se è un indirizzo di gruppo multicast; in tal caso si passa al punto 6;
5. scarta il pacchetto e notifica il mittente che l'indirizzo di destinazione non è noto;
6. consulta la tabella di instradamento per decidere su quale porta di uscita rilanciare il pacchetto.

Infine, un esempio di uso di temporizzatori si ha nel caso di controllo della corretta ricezione di un messaggio effettuato mediante riscontri: una entità A invia un messaggio a B ed attende da B una esplicita notifica di ricezione corretta del messaggio. Quando deve aspettare A? Come fa a sapere che "è troppo tardi"? Una risposta accurata può essere anche molto difficile da dare, per esempio nel caso di ritardo di trasferimento tra A e B variabile da pacchetto a pacchetto. È però chiara l'esigenza di mettere un termine all'attesa del riscontro da parte di A per evitare di mandare il protocollo in uno stato di stallo. La soluzione adottata universalmente è definire un "time-out", cioè un contatore che scade dopo un tempo prefissato a partire dall'invio dei dati. Se entro questa scadenza non si è ricevuto riscontro, si assume che i dati non siano giunti a destinazione. Porre una terminazione certa all'attesa del riscontro è un requisito esattamente analogo a quello di terminazione di un algoritmo. Con la difficoltà enorme del fatto che l"algoritmo" in questione (eseguito dal protocollo) è intrinsecamente di natura distribuita e prevede la collaborazione di sistemi di elaborazione remoti.

rio ritornare all'ultima situazione concordata, annullando tutte le situazioni pendenti (processo di uccisione degli orfani). Un esempio classico è costituito dal processo collaborativo per cui si aggiornano contemporaneamente più basi di dati distribuite. L'algoritmo utilizzato implica una regia piuttosto complessa in più fasi (*prepare, commit, rollback*); in caso di problemi si torna all'ultima condizione certa (*rollback*).

In altre situazioni è necessario ricorrere ad algoritmi "a latere", molto sofisticati e apparentemente scollegati dal processo in esame. Un esempio semplice, ma concreto, è costituito dalla banale richiesta di conferma della ricezione di un documento da parte del destinatario (si intende qui la ricezione effettiva, ossia la verifica che il documento sia stato effettivamente aperto sul PC del destinatario e non il più semplice problema della consegna "postale"). La certezza dell'effettivo display del documento si ottiene con algoritmi complessi di tipo "notarile" (ovvero è necessario l'intervento di un garante "esterno" che esegua ogni richiesta in duplice formato: in chiaro e "crittografata" con la chiave pubblica del mittente in modo che si possa sempre dimostrare che non ci sono discrepanze tra i documenti che il destinatario ha accettato di ricevere e quelli che gli vengono effettivamente consegnati).

Nel seguito dell'articolo, dopo una breve storia dei protocolli (paragrafo 2), sono presentate le idee di base ormai accettate e consolidate nelle architetture di protocolli di comunicazione. In questo contesto sono introdotte le architetture di comunicazione e si chiarisce il nesso con l'aggiunta di valore, caratteristica saliente di un vero protocollo (almeno di quelli utili). È anche brevemente discusso l'impatto sulle architetture di protocolli di requisiti trasversali: la qualità di servizio, la sicurezza nella comunicazione e il risparmio energetico. Spazio è dedicato nel paragrafo 4 a discutere il nesso tra il modello architetturale introdotto nel paragrafo 3 e la pila protocollare di Internet. Il paragrafo 5 contiene uno sguardo ad approcci "alternativi", in particolare al *cross-layering*, per il loro interesse negli sviluppi di specifiche importanti tecnologie come il *wireless*.

Esempi tratti dalle principali architetture di protocolli sono diffusi nel testo nel tentativo di rendere afferrabile ogni definizione astratta e di mostrare come si concretano le soluzioni a problemi di comunicazione in rete attraverso meccanismi protocollari.

Spesso questi contengono sottigliezze e dettagli che richiedono profonde conoscenze e riflessione attenta per essere comprese; ma altrettanto spesso, procedure protocollari infarcite di sigle e di tecnicismi sono il buon senso codificato!

2. UNA BREVE STORIA

I primi protocolli nascono per risolvere il problema della trasmissione remota di dati, intrinsecamente legato al problema della condivisione di una risorsa, ai tempi preziosissima, quale la banda trasmissiva e la capacità di elaborazione.
Normalmente sono noti con il nome di "protocolli orientati al carattere" [1, 3] in quanto il meccanismo per gestire la "regia" del dialogo consiste nell'eleggere alcuni caratteri a funzioni di controllo: per esempio un carattere rappresenta l'inizio del blocco di dati (STX, *Start of Text*), un altro la fine (EOT, *End of Transmssion*), un altro ancora la conferma di ricezione corretta – o meno – (ACK o NACK, *Acknowledgement o Not Acknowledgement*).
Questi protocolli sono quasi sempre utilizzati su una linea multipoint (ovvero su una linea condivisa da più dispositivi), per cui é necessario introdurre un meccanismo selettivo di trasmissione/ricezione (un meccanismo possibile é il *polling/selecting*, ovvero l'appello a turno ciclico per autorizzare la trasmissione).
I limiti di questi protocolli sono moltissimi. Ne elenchiamo i principali:

- ❑ non trasparenza al testo. Il testo non può contenere caratteri codificati come uno dei caratteri di controllo (per evitare ambiguità). Il problema si supera con un ulteriore livello di caratteri di controllo (DLE, *Data Link Escape*, per segnalare che il carattere successivo va inteso come testo e non come controllo);
- ❑ non trasparenza al controllo. L'inserimento di un nuovo carattere di controllo (per implementare una nuova funzione), può far insorgere ancora problemi di trasparenza;
- ❑ assenza di controllo di flusso: non é possibile inviare un nuovo blocco di informazioni se il ricevente non ha dato l'ACK a quello precedente (al massimo si può pensare a due blocchi, con una logica di ACK pari e dispari);
- ❑ ambiguità nei caratteri di controllo. Spieghiamo con un esempio: un ACK può essere interpretato a livello trasmissivo, oppure applicativo, oppure transazionale (ho ricevuto il blocco, l'ho ricevuto e l'ho passato all'applicazione, l'applicazione l'ha preso in carico ed ha logicamente chiuso la transazione). Questo problema é in assoluto il più grave in quanto rende il dialogo fortemente dipendente dal contesto in cui é inserito, a livello fisico e logico;
- ❑ povertà del controllo di errore, basato sui bit di parità;
- ❑ impossibilità di gestire (orchestrare) più flussi contemporanei.

Un passo importante nell'evoluzione dei protocolli é costituito dall'introduzione (da parte di IBM) dell'SDLC (*Synchronous Data Link Control*, [2]), da cui é derivato l'importantissimo (e tuttora vivo) standard HDLC (*Higher-Level Data Link Control*, [4]). HDLC e SDLC sono strutturalmente identici, ma sono utilizzati in contesti diversi. Non é importante capirne le differenze, ma é importante ragionare sulla novità introdotta.
Essi sono chiamati "protocolli orientati al bit" (in contrapposizione a quelli precedenti, orientati al carattere), perché si supera il concetto di carattere di controllo, ma si introduce quello di "trama" o "cornice". Ovvero la busta trasmissiva ha una struttura predefinita - inizio, campo indirizzo, caratteri di controllo, testo, controllo di errore, fine - per cui una sequenza di bit ha un significato per la posizione occupata all'interno della trama e non solo per la sua codifica. L'*n*-simo byte sarà sempre e solo un carattere di controllo (o di testo), in relazione alla sua posizione, e così sarà comunque interpretato.
Con i protocolli orientati al bit si superano enormi problemi.
Il testo é trasparente ed indipendente dalla codifica, a patto di rendere univoca la delimitazione delle trame (per esempio con meccanismi di *bit stuffing*).
Il campo indirizzo permette di indirizzare un numero elevato di stazioni, lavorando su linee punto a punto oppure su linee multipunto.
Il campo controllo contiene molteplici funzioni; una delle più importanti é la numerazione dei blocchi in trasmissione e ricezione, secondo un modulo aritmetico predefinito (per esempio 8 o 128). I numeri in trasmissione individuano la sequenza dei

blocchi in trasmissione, i numeri in ricezione individuano il limite superiore dei blocchi ricevuti correttamente nel verso opposto. Questo meccanismo (chiamato finestra) permette di avere un flusso trasmissivo regolabile, basato su meccanismi semplici e potenti di controllo di flusso e di errore. La regolazione del flusso riguarda ogni aspetto trasmissivo: sono infatti risolti problemi di sequenzialità e di integrità (per effetto della numerazione), di regolazione della quantità di informazioni in transito (si può arrivare a spegnere completamente il dialogo in una direzione, non aggiornando il limite inferiore della finestra e quindi esercitare un completo controllo di flusso oppure dargli il massimo dell'apertura aggiornando continuamente il limite inferiore della finestra in trasmissione), di regolazione del turno (il flusso può essere mono o bidirezionale, essendo i campi di controllo presenti anche in un "frame" di tipo informativo).
Il controllo di errore é basato su meccanismi CRC (*Ciclic Redundancy Check*) – ovvero sul calcolo del resto utilizzando il testo come dividendo e un polinomio standard come divisore; questo algoritmo rende la trasmissione praticamente immune da errore (la probabilità di avere due testi differenti con il medesimo resto é praticamente nulla).
I protocolli orientati al bit risolvono quindi brillantemente il problema della trasmissione tra due entità. Lasciano però, purtroppo, aperti due problemi essenziali:

- l'impossibilità di distinguere tra elementi di controllo trasmissivo ed elementi di controllo applicativo (l'ACK é ancora ambiguo – resistono ancora gli stessi problemi dei protocolli orientati al carattere);
- l'impossibilità di eseguire controlli selettivi all'interno del flusso trasmissivo (il non-aggiornamento del limite inferiore della finestra determina il blocco completo a livello trasmissivo, senza la possibilità di operare in modo selettivo su eventuali differenti flussi applicativi in essere tra i due interlocutori).

Contestualmente, dalla fine degli anni '60, sono stati messi a punto protocolli per il trasferimento dei dati a pacchetto in reti geografiche. Alcune delle idee più feconde sono maturate con un lento processo di *trial & error* nell'ambito del progetto ARPANET, il progenitore di Internet (vedi http://www.isoc.org/internet/history/; [5]).
I problemi della comunicazione di dati e la strutturazione dei relativi protocolli hanno portato nei primi anni '80 al maturare del cosiddetto modello OSI, oggetto del paragrafo seguente.

3. LE IDEE DI BASE DELLA MODERNA CONCEZIONE DEI PROTOCOLLI

Progettare e realizzare un protocollo è ad oggi ancora un'arte, come alcuni amano dire, o forse più appropriatamente, un'opera di artigianato. Questo non toglie che nel tempo sono stati sviluppati modelli e strumenti per rendere sistematico e verificabile il lavoro di definizione e sviluppo di un protocollo.
Il più famoso al riguardo è il modello OSI (*Open System Interconnection*), emesso come Standard ISO 7498 nel 1983 e recepito quindi come Raccomandazione ITU X.200. Il modello si propone di definire una architettura funzionale completa per descrivere la cooperazione di processi remoti, residenti cioè in sistemi distinti e comunicanti tra loro, in altre parole in grado di trasferire stringhe binarie con date caratteristiche di prestazione in termini di capacità e affidabilità. Punti chiave del modello sono la decomposizione del complesso delle funzioni di comunicazione e di elaborazione locale dell'informazione in sotto-insiemi strutturati gerarchicamente e la formalizzazione delle interazioni e dei flussi informativi nelle varie interfacce identificate nell'architettura. Questi aspetti sono l'eredità ancora viva del modello OSI, perfettamente identificabili e anzi alla base dello sviluppo di standard complessi relativi a qualsiasi tipo di rete. È quindi su questi aspetti che sarà posto l'accento nel seguito; per una descrizione esauriente del modello OSI si possono consultare, oltre agli standard, che ne sono la fonte primaria, molti lavori (ottima è la descrizione del modello OSI contenuta nella documentazione tecnica Cisco [6]).
L'idea di base del modello è che l'eterogeneità e la vastità delle funzioni coinvolte nel processo di cooperazione è affrontabile al meglio solo suddividendo queste funzioni in

gruppi omogenei per livello di astrazione del trattamento delle informazioni e per affinità tecnologica di realizzazione.
L'equalizzazione del canale trasmissivo, il recupero del cronosegnale associato al segnale numerico ricevuto, la rivelazione di errori nei blocchi di informazione trasferiti, l'instradamento dell'informazione tra i sistemi terminali dello scambio informativo attraverso eventuali sistemi intermedi, la transcodifica dell'informazione tra formati diversi di rappresentazione (per esempio, da mp3 a wav per un file audio; oppure trancodifiche tra i diversi formati adottati per la voce nelle reti cellulari o nell'interconnessione tra reti telefoniche e *VoIP*) sono tutti esempi di funzioni generalmente eseguite in ogni singola istanza di cooperazione tra processi remoti e chiaramente eterogenee tra loro. Le prime due funzioni possono essere realizzate tipicamente con hardware o firmware e considerano l'informazione trasferita tra due sistemi adiacenti (senza sistemi intermedi) come un semplice flusso di simboli, per esempio binari. La rivelazione di errori è realizzabile solo articolando l'informazione in blocchi delimitati e strutturati ed è spesso realizzata in software microprogrammato su schede dedicate. L'instradamento è una funzione di natura logica, che coinvolge più sistemi, richiede l'esecuzione di algoritmi e procedure tipicamente realizzati in software all'interno del *kernel* del sistema operativo e tratta informazioni assunte esenti da errori e strutturate in messaggi indirizzabili. Infine, la transcodifica si applica a dati di utente, è eseguita localmente dal processo che ha ricevuto l'informazione, assunta esente da errori e ricostruita nella sua interezza così come la sorgente l'ha generata.
Anche dagli esempi appena accennati è evidente che si possono individuare gruppi di funzioni separati, la cui esecuzione implica l'interazione di sistemi diversi, o meglio, di quelle parti dei sistemi che presiedono al gruppo funzionale considerato. È anche chiaro che le funzioni così raggruppate non sono svolte in un ordine qualsiasi, ma ordinate in modo gerarchico. Per il destinatario dell'informazione e con riferimento agli esempi fatti sopra, occorre prima rivelare il flusso di informazione ricevuto, eseguendo tra le altre l'equalizzazione e il recupero della sincronizzazione, quindi delimitare blocchi di informazione significativi e accertarsi della loro correttezza, recuperando eventuali errori, e così via.
Nasce quindi il fondamentale concetto di *strato*. Uno strato dell'architettura di comunicazione è l'insieme delle *entità* appartenenti a tutti i sistemi che svolgono un dato gruppo di funzioni. Le entità in concreto sono codice, hardware, schede; la loro cooperazione avviene mediante *protocolli*, che hanno appunto lo scopo di svolgere le funzioni relative allo strato cui appartengono le entità. Dal concetto di strato e di stratificazione delle funzioni nasce il valore aggiunto: ogni strato aggiunge "valore" al processo di cooperazione tra i processi remoti, mediante l'azione svolta dalle proprie entità nell'esecuzione dei protocolli cui sono preposte. È per esempio valore aggiunto il poter garantire l'affidabilità del trasferimento informativo ad un livello desiderato data un'infrastruttura fisica di per se non soddisfacente i requisiti. Oppure la capacità di recapitare al corretto destinatario l'informazione che gli compete, permettendo in ultima analisi l'interconnessione mediante una rete con condivisione delle risorse, piuttosto di una connessione diretta fisica, con risorse dedicate, tra ogni coppia di sistemi che debbano cooperare.
Il meccanismo di comunicazione tra processi in un'architettura stratificata è accuratamente descritto dal modello OSI, ma può forse essere esemplificato e compreso nella sua essenza con un colorito esempio, ispirato da Andrew S. Tanenbaum [1, paragrafo 1.3]. Un filosofo cinese e un bramino indiano desiderano dialogare; dato che parlano lingue diverse, ognuno dei due ingaggia un interprete, concordando una lingua comune (dal mandarino all'inglese uno, dal sanscrito all'inglese l'altro). Poiché i due interpreti a loro volta si trovano uno a Pechino, l'altro a Benares, essi ingaggiano un telegrafista ognuno e incaricano i telegrafisti di trasmettere i messaggi originati dai filosofi e tradotti in inglese. È chiaro che il bramino che esprime un pensiero sta (virtualmente) dialogando con il filosofo cinese e NON con il proprio interprete; per attuare questo dialo-

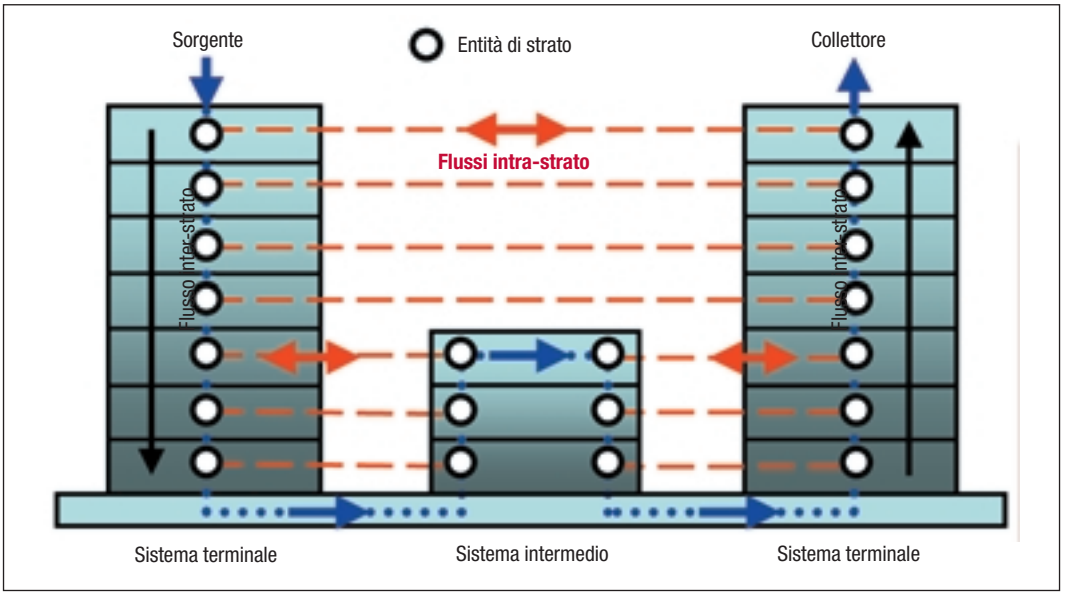

FIGURA 1
Trasferimento dell'informazione in un'architettura a strati

go tuttavia, egli affida fisicamente il proprio messaggio all'interprete, il quale a sua volta dialoga (virtualmente) con l'altro interprete; per far questo egli affida fisicamente il messaggio al telegrafista che lo trasmette tramite un mezzo fisico di comunicazione al telegrafista remoto. Approdato qui, il messaggio risale la "pila" in senso inverso fino al filosofo. Il paradigma di comunicazione è quindi "orizzontale" virtualmente, cioè i protocolli di comunicazione e cooperazione sono eseguiti tra *entità alla pari* (*peer entities*). È "verticale" nelle modalità di attuazione, cioè le entità di uno strato (dette appunto "utenti") si affidano a quelle dello strato gerarchicamente inferiore (dette "serventi") per supportare il proprio dialogo (Figura 1).
L'OSI individua sette strati (Figura 2). I quattro più in basso, in ordine Fisico (*Physical*), di Collegamento (*Data Link*), di Rete (*Network*) e di Trasporto (*Transport*) sono relativi alle funzioni di trasferimento dell'informazione. I tre più in alto, Sessione (*Session*), Presentazione (*Presentation*) e Applicativo (*Application*), sono relativi al trattamento locale dell'informazione. Questa particolare suddivisione non è che una tra le possibili (e certo non tra quelle di maggior successo). Il concetto di strato funzionale è invece diventato pervasivo e costituisce il mattone base di tutte le architetture di comunicazione concepite dagli anni '80 in poi (riquadro a p. 28).

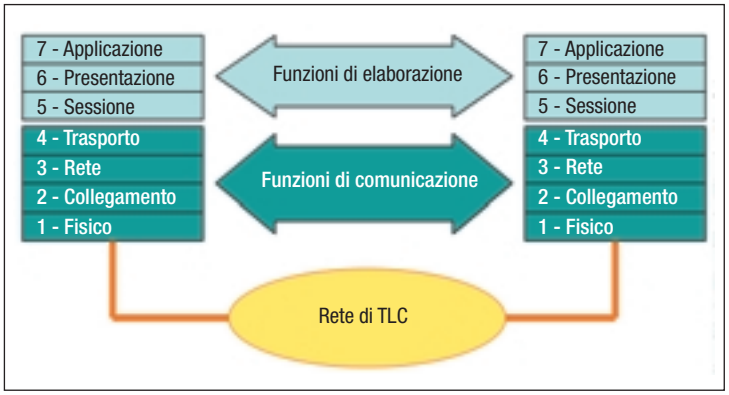

FIGURA 2
Schema dell'architettura OSI

Il modello introduce una descrizione generale degli strati e dei loro elementi componenti, riferendosi ad un generico (*N*)-strato, con *N* intero positivo.
Un elemento basilare dei protocolli è la definizione delle unità informative scambiate dalle entità di un dato strato, le così dette (*N*)-PDU (*Protocol Data Unit* dello strato *N*). Le trame di un protocollo MAC, i segmenti del TCP, i pacchetti IP sono tutti esempi di PDU. Il contributo del modello OSI sta nell'aver formalizzato gli elementi comuni di cui ogni PDU è formata, astraendo dai dettagli implementativi:
- una PCI (*Protocol Control Information*), cioè l'insieme delle informazioni di controllo che occorrono perché la PDU concorra all'assolvimento delle funzioni del protocollo;

I sette livelli proposti nel modello OSI

Livello "1": il livello fisico. É responsabile della codifica del segnale a livello elettrico, delle procedure di "handshaking" per la creazione e il mantenimento dei canali trasmissivi e anche delle specifiche strutturali dei connettori e dei mezzi di trasmissione. Le sue funzioni variano secondo la natura del canale trasmissivo.

Livello "2" il livello di collegamento. É responsabile della correttezza e della integrità dei dati trasmessi. I meccanismi utilizzati sono quelli descritti nel paragrafo 2.

Livello "3": il livello di rete. É responsabile dell'instradamento delle informazioni. É uno dei livelli più complessi e può essere "connesso" o "non connesso": un protocollo connesso invia dati al sistema terminale (DTE) destinatario solo dopo una fase di negoziazione in cui i due DTE concordano sulla modalità di trasmissione (dal banale "essere pronti a ricevere" ad una complessa negoziazione sulla qualità del servizio). Ci sono ovviamente vantaggi (e svantaggi) in ognuna delle due soluzioni. Esempi di protocolli connessi sono l'X.25 e l'ATM; un esempio universale di protocollo non connesso é l'IP di Internet (per inciso, questa affermazione ci permette di capire perché Internet non é nativamente adatta a gestire connessioni in cui é richiesta una qualità garantita, ad esempio la voce!).

Livello "4": il livello di trasporto. É il primo livello di dialogo tra i due DTE ed é responsabile della correttezza della trasmissione da estremo a estremo (end-to-end). Notiamo che il livello "2" e "3" hanno un significato locale: solo i due interlocutori remoti possono infatti avere il controllo completo (e inconfutabile) della correttezza della trasmissione. Ciò é evidente in caso di anomalie della rete (perdita di blocchi di informazione). Le funzioni del livello "4" variano in funzione della qualità della rete di trasporto. Altre importanti funzioni di questo livello sono il multiplexing (ovvero l'affasciamento contemporaneo di più connessioni eventualmente in essere tra i due DTE remoti), il controllo di congestione e il controllo di flusso.

Livello "5": il livello di sessione. Cominciamo ad essere vicini al mondo applicativo. Il livello "5" si occupa degli strumenti che servono alla gestione del dialogo applicativo (che può essere di natura estremamente varia: ad esempio un'applicazione che pilota una stampante remota, oppure due programmi che cooperano per eseguire una transazione congiunta). Il livello di sessione definisce ad esempio i turni di dialogo, mette delle "pietre miliari" per permettere ai programmi di sincronizzarsi su punti concordati, ad esempio in caso di anomalie nel flusso trasmissivo. É importante sottolineare che il significato di questi "paletti" é definito dall'applicativo che li utilizza; ancora dunque, come in tutta l'architettura OSI, essi sono un servizio fornito al livello superiore (l'applicazione, in questo caso) e dunque non hanno un significato assoluto.

Livello "6": é il livello di presentazione. Potrebbe essere inglobato nel livello di sessione. É responsabile della codifica e della presentazione dell'informazione.

Livello "7": il livello applicativo. Qui si possono ingenerare delle confusioni. Il livello applicativo infatti non definisce le "applicazioni", ma i servizi applicativi. Servizio applicativo é ciò che serve all'applicazione finale (il processo di business) per lavorare in un ambiente aperto e distribuito. Un esempio di servizio applicativo universalmente utilizzato é la posta elettronica (non é essa stessa un'applicazione, essendo l'applicazione il client di posta che permette di comporre il messaggio o l'applicazione verticale che muove informazioni usando i meccanismi di posta).

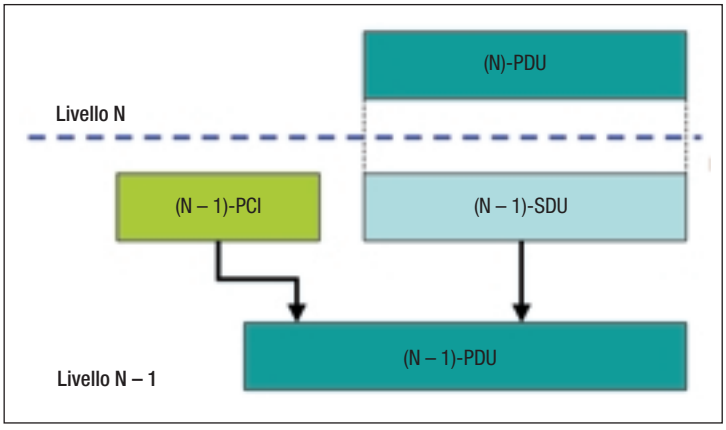

FIGURA 3
Struttura delle unità informative nel modello OSI

- una SDU (*Service Data Unit*), opzionale, parte dedicata al trasporto dei dati offerti dallo strato superiore (Figura 3).

È importante osservare che la definizione di (*N*)-PDU come unione di una (*N*)-PCI e di una (*N*)-SDU è ricorsiva, nel senso che a sua volta la (*N*)-SDU corrisponde (nel caso più semplice) ad una (*N* + 1)-PDU. Il punto di vista dello strato *N* sulla (*N*)-SDU è che essa contiene i dati del livello superiore (l'utente, in senso OSI naturalmente), affidati per il trasferimento alle entità dello strato *N* e trattati come una "scatola nera"; il protocollo dello strato *N* si impegna a effettuare il trasferimento consegnando i dati di utente alle entità di destinazione dello strato *N* + 1, dopo aver aggiunto il proprio "valore" (per esempio, garantito l'assenza di errori e di fuori sequenza entro un fissato margine di probabilità di successo). Gli stessi dati, visti dal livello *N* + 1, sono invece un gruppo strutturato di bit rappresentanti un messaggio dell'(*N* + 1)-protocollo (una (*N* + 1)-PDU), che può a sua volta essere separato in una parte che contiene informazioni di controllo, la (*N* + 1)-PCI, e in una eventuale che contiene i dati del livello superiore *N* + 2, la (*N* + 1)-SDU.

Può sembrare un modello astratto, artificiale e non corrispondente alla multiforme realtà dei protocolli reali. In effetti, è una delle chiavi interpretative e di razionalizzazione nella definizione e sviluppo di protocolli più utili

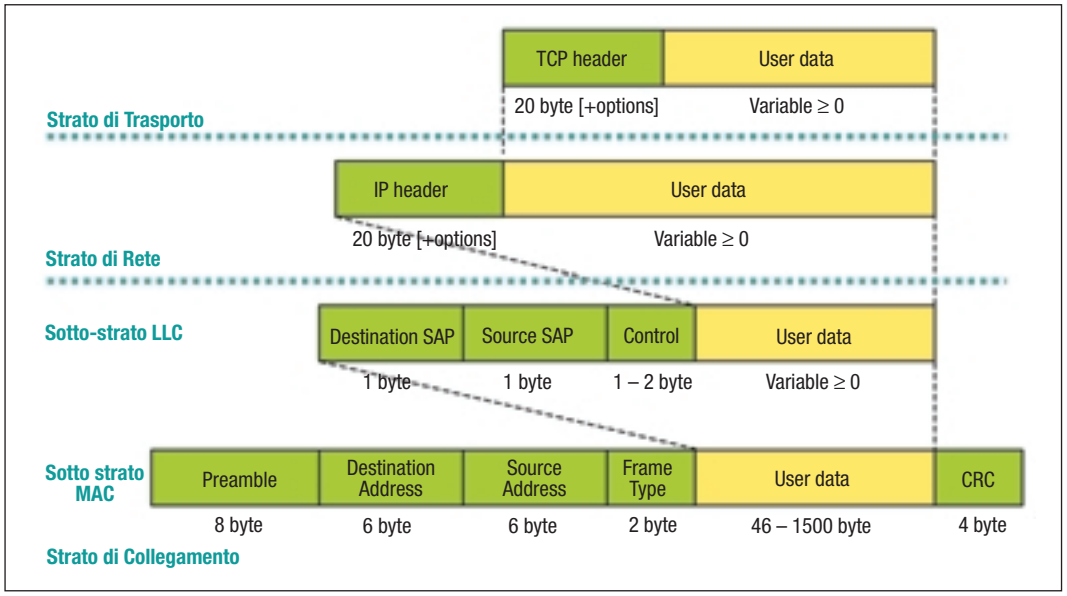

FIGURA 4

Esempio di relazione tra PDU: dall'alto verso il basso sono mostrati un segmento TCP, un pacchetto IP, una trama LLC e una trama Ethernet

tra quelle fornite dal modello OSI. L'approccio a scatole cinesi, indipendenti nella definizione delle procedure interne, è l'elemento abilitante per uno sviluppo modulare e interoperabile di sistemi complessi, che vanno spesso soggetti nel corso della loro vita ad aggiornamenti tecnologici.

Un esempio di annidamenti di PDU e aggiunta di informazioni di controllo da parte di entità di strati successivi è mostrato nella figura 4. Per ogni strato la relativa PDU è esplicitamente mostrata con la sua informazione di controllo (in verde) e la parte che ospita i dati dello strato superiore (in giallo). Ogni strato inserisce nella propria PCI quello che serve alle funzioni del relativo protocollo di strato (indirizzi, campi di versione, di lunghezza e qualificazione delle parti che seguono, numerazione per il controllo di sequenza, riscontri, flag per gestire la consegna dei dati, la loro eventuale frammentazione, campi di controllo di errore ecc.).

Una PDU può ridursi alla sola PCI: con riferimento all'esempio in figura 4, i segmenti TCP usati dal destinatario dei dati come riscontro sono tipicamente (ma non necessariamente) ridotti alla sola intestazione di 20 byte. In generale, un protocollo si svolge attraverso lo scambio di PDU tra le entità dei sistemi coinvolti. Le PDU assolvono compiti di controllo (per esempio quelli richieste nel caso di instaurazione di una connessione preliminarmente all'invio dei dati), di gestione (per esempio PDU per verificare la presenza dell'entità remota o per effettuare misure), di trasporto dei dati di utente. Nei protocolli moderni le PDU sono stringhe binarie, strutturate in "campi", cioè composte di sequenze di bit che sono interpretate in funzione della posizione che esse occupano nell'ambito della PDU, del proprio contenuto binario e di altri campi della PDU. È fondamentale rammentare che i bit sono bit e la rappresentazione di una PDU come composta da campi con i propri nomi in bell'ordine, come nella figura 4, è solo nella mente di chi concepisce il protocollo e non è "visibile" al codice o all'hardware che lo esegue. È quindi fondamentale garantire la possibilità di *delimitare* le PDU e quindi di scomporle in modo univoco nei campi costituenti, comprese le opzioni e le varianti possibili (questa funzione si chiama *parsing*).

Il formato delle PDU, sia esso specificato mediante rappresentazione dei campi componenti come nella figura 4 oppure con metodi più sofisticati quali ASN.1 (come accade in molti esempi di protocolli soprattutto di livello applicativo), permette di descrivere gli elementi con i quali realizzare il servizio offerto dal protocollo.

Un protocollo di strato *N* offre servizio alle (*N* + 1)-entità; esse possono stimolare le entità del livello servente (lo strato *N*) mediante "primitive" di servizio (Figura 5): per esempio, il trasferimento di una PDU di dati di utente avviene emettendo una primitiva di *Data-Request*, nella quale si passano come parametri quella che sarà la (*N*)-SDU e parametri aggiuntivi (parametri di QoS, indirizzi di destinazione, specifiche sulle modalità di trattamento dei dati da parte dell'(*N*)-protocollo, per esempio richiesta di cifratura). Il protocollo dello strato *N* effettua il trasferimento dei dati con tutti i meccanismi che gli sono propri (per esempio quelli di rivelazione e recupero degli errori), e presenta infine una primitiva di Data-Indication alla (*N* + 1)-entità destinataria.
In concreto le primitive possono essere segnali su circuiti interni a una macchina, interrupt (software o hardware), chiamate a funzioni o procedure nell'esecuzione di un programma, sia del sistema operativo sia di un processo di area utente, a seconda del livello architetturale degli strati cui le primitive si riferiscono. Questa enorme varietà di modalità realizzative viene ricondotta a concetti semplici e unitari, di grande valore nel concepire, definire e sviluppare in concreto i protocolli.
La razionalizzazione del funzionamento di un sistema inserito in una rete di comunicazione si è dimostrata feconda. Non c'è standard di sistemi di comunicazione che non descriva le funzioni svolte strato per strato e quindi le primitive per quanto riguarda le interfacce "verticali" e le procedure protocollari per quelle "orizzontali", cioè tra entità alla pari in sistemi remoti.
Altri due concetti fondamentali e ricorrenti nei sistemi di telecomunicazione sono la *multiplazione* (con la corrispondente funzione inversa di *demultiplazione*) e la *connessione*.
In termini OSI un protocollo di strato N effettua multiplazione quando si pone al servizio di una molteplicità di (*N* + 1)-entità (gli utenti del servizio erogato dal (*N*)-protocollo) e trasporta quindi con le proprie (*N*)-PDU flussi di dati appartenenti a diverse relazioni tra entità di strato *N* + 1. Un esempio è il protocollo IP che trasporta segmenti appartenenti a diverse connessioni di trasporto TCP, datagrammi di diversi flussi UDP e unità di dati di molti altri protocolli. Un aspetto cruciale, immediatamente ovvio dal modello astratto della multiplazione è che deve essere inserita nella (*N*)-PCI informazione sufficiente a identificare a quale (*N* + 1)-entità è destinata una data (*N*)-SDU. Nell'esempio del pacchetto IP, l'informazione in questione è l'indirizzo IP dell'interfaccia di destinazione del pacchetto e il campo Protocol Type che specifica a quale modulo software del destinatario va consegnato il contenuto di dati del pacchetto IP.
La connessione è un'associazione logica tra due o più entità remote che permette loro di istanziare e mantenere aggiornati coerentemente i rispettivi stati di evoluzione nell'esecuzione di un protocollo di strato. Il concetto di connessione accoppiato con quello di architettura stratificata si rivela molto potente: esso permette di evidenziare in modo chiaro e di razionalizzare le interdipendenze tra sistemi e tra diversi compiti svolti nel processo di comunicazione. Interagire con o senza connessione è lo spartiacque tra la possibilità di supportare qualità di servizio negoziata, affidabilità del trasferimento dei dati, e

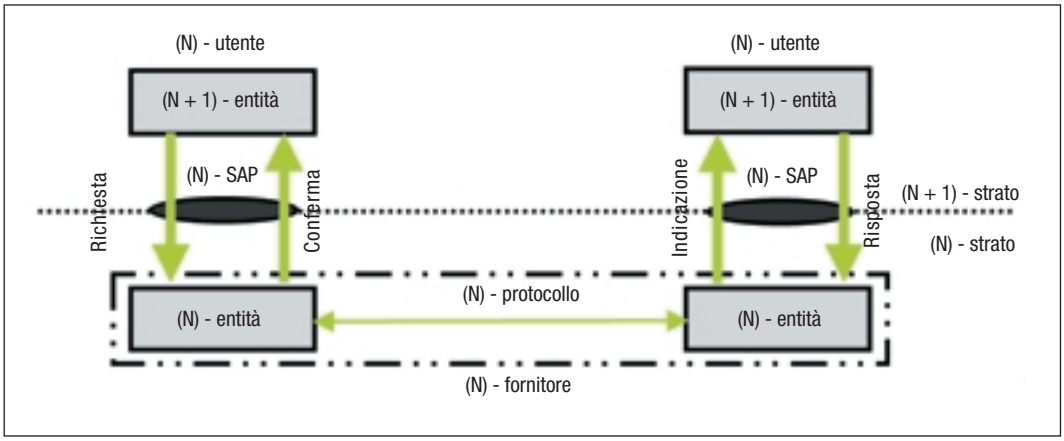

FIGURA 5
Interazioni tra strati adiacenti per la fornitura di un elemento di servizio secondo il modello OSI

molti altri arricchimenti del servizio offerto da un protocollo e la semplicità di realizzazione e quindi la parsimonia di risorse di memoria e calcolo richieste. Per mantenere in sequenza completa i diversi frammenti di dati trasferiti nell'ambito di un'interazione tra entità remote, controllare il flusso o garantire un fissato grado di servizio (ritardo per esempio) sono necessari contatori e meccanismi di recupero dei dati mancanti. Questo è possibile instaurando preventivamente una connessione (quindi inizializzando i contatori in modo coerente). D'altra parte questo richiede buffer e esecuzione di molte istruzioni in entrambi i "capi" della connessione. Inoltre rende complicato adattarsi a variazioni del servizio offerto dagli strati sottostanti.
È importante osservare che, coerentemente con il principio dell'indipendenza tra strati, la modalità di funzionamento con o senza connessione di un (*N*)-protocollo non influenza quelle dei protocolli degli strati adiacenti. Prendiamo il colloquio tra un PC di utente privato e il web server del suo ISP. Lo strato fisico tra PC e modem ADSL è tipicamente realizzato con interfacce USB (con connessione) ovvero Ethernet (senza connessione). Al di sopra si usa il PPP (con connessione) o il MAC Ethernet (senza connessione). Tra modem ADSL e NAS si usa una connessione ATM (strato due). Per lo strato di rete, il terzo, si usa IP tra PC e web server (senza connessione), al di sopra si usa TCP (con connessione) e al di sopra ancora HTTP (senza connessione).
Il vantaggio della modalità a connessione è la potenziale ricchezza di funzioni eseguibili solo grazie all'associazione logica stabilita tra le parti con la connessione; il vantaggio della modalità senza connessione è la semplicità delle entità che eseguono il protocollo[1]. Non esiste una soluzione migliore dell'altra. In ogni sistema e strato l'una o l'altra sono le più indicate, dato il contesto. Molti protocolli di strato due sono realizzati con la possibilità di optare per l'una o l'altra modalità, in funzione degli scopi e delle risorse che le entità coinvolte possono dedicare all'interazione.

[1] Una diffusa realizzazione (4.4BSD-Lite distribution) di TCP e IP (il primo con connessione, il secondo senza) consta di circa 15000 e 2000 righe di codice rispettivamente.

Importanti valori aggiunti dai protocolli di comunicazione riguardano aspetti trasversali dal punto di vista architetturale, nel senso che il loro trattamento non può essere confinato in generale ad un ben determinato strato. Se ne citano qui tre: la *Qualità di Servizio* (QoS, *Quality of Service*), la sicurezza (nel senso del termine anglosassone *security*, da non confondere con *safety*), il risparmio energetico.
La *QoS* si concreta in metriche di prestazione specifiche dello strato nella quale si considera: può riferisce al rapporto segnale-rumore o *BER* (*Bit Error Ratio*) di un collegamento trasmissivo nello strato fisico, alla portata media e al ritardo di una connessione nello strato di trasporto, alla flessibilità d'uso del protocollo da parte dell'utente nello strato applicativo. La "portata" della *QoS* è legata allo strato al quale essa è misurata: si tratta ovviamente di *QoS* da estremo a estremo negli strati alti, di *QoS* locale per gli strati bassi.
Il progetto di un protocollo è sempre la risposta a un'esigenza di *QoS*: gli algoritmi che ne sono alla base, eseguiti dalle entità coinvolte nel protocollo hanno sempre obiettivi prestazionali che mirano ad aggiungere funzionalità al trasferimento di segnali permesso da mezzi fisici di comunicazione allo scopo finale di supportare la cooperazione di processi applicativi più o meno sofisticati.
Gli aspetti di sicurezza sono anch'essi trasversali nel senso che possono essere introdotti in protocolli di tutti gli strati: si pensi alla cifratura a livello fisico, alla realizzazione di tunnel sicuri al livello rete, ai processi di autenticazione al livello di collegamento o applicativo tipici ormai della maggior parte delle nostre interazioni in rete (dall'accesso alle reti cellulari o WiFi, alle connessioni PPP via modem, all'accesso a porzioni riservate di siti web o serventi di posta elettronica, ad applicazioni di commercio elettronico). Basti qui osservare che l'introduzione di funzioni di sicurezza (confidenzialità, autenticazione, integrità, non-ripudio, disponibilità) può tradursi in aggiunta di procedure a protocolli esistenti, magari attraverso campi opzionali o nuove PDU definite nel protocollo, oppure può tradursi nella definizione di veri e propri nuovi protocolli, il cui precipuo obiettivo di arricchimento di valore è offrire servizi di sicurezza nell'am-

bito dell'interazione tra utenti e/o processi.
Infine, considerazioni di dissipazione di energia nella comunicazione diventano importanti quando almeno alcuni dei sistemi partecipanti alle interazioni protocollari sono alimentati a batteria, come nelle reti cellulari e wireless (WLAN, WPAN, reti di sensori). È intuitivo che i protocolli di livello fisico siano interessati nella gestione dell'energia (si veda per esempio la funzione di controllo di potenza), ma non meno di questi sono coinvolti nel risparmio energetico anche i protocolli degli strati di collegamento, rete e trasporto. Due esempi per chiarire. Il recupero di errori mediante ritrasmissione ha un costo energetico; l'urgenza e la stessa opportunità di ritrasmettere una PDU di dati va quindi bilanciata con il costo energetico che questo comporta. Per sistemi che prevedano un accesso multiplo coordinato da un apposito protocollo (si chiamano protocolli MAC, *Medium Access Control*), come la maggior parte dei sistemi di accesso wireless, le procedure di accesso prevedono spesso la funzione di ascolto del mezzo trasmissivo per evitare o almeno limitare le collisioni; una scheda wireless consuma una quantità di energia quasi uguale sia quando è in trasmissione sia quando è in ascolto[2]. È chiaro come anche l'aspetto energetico deve essere valutato nella definizione di protocolli che debbano funzionare in sistemi alimentati a batteria.

Come rinunciare alla connessione e mantenere uno stato: HTTP e i cookies

Nel caso di un server applicativo che deve contemporaneamente trattare numerose richieste di servizio da parte di una popolazione di client può essere molto oneroso inizializzare e far evolvere uno stato per ogni singola comunicazione, Queste considerazioni hanno per esempio portato a definire il protocollo alla base del World Wide Web , l'HTTP, prevedendo una modalità di comunicazione senza connessione (si dice che HTTP è *stateless*). Ogni interazione di un client con un server web si riduce ad uno schema query-response: il client chiede un contenuto, il server lo rintraccia, lo invia e ...si può dimenticare del client!
Per estendere le capacità dei server web, ferma restando la natura stateless di HTTP, sono stati introdotti i cosiddetti *cookies*. Un server impacchetta lo stato riguardante l'interazione con il client che lo ha contattato e lo invia al programma client, il quale lega il *cookie* all'URL del sito che glielo ha inviato. Alla successiva interazione indirizzata a quell'URL da parte di quel client, il cookie viene inviato al server, così "rammentandogli" lo stato della precedente interazione. Un bell'esempio di come mantenere la semplicità del paradigma di comunicazione connectionless e ottenere comunque una "memoria", magari con qualche rischio per la sicurezza.

4. UN CONFRONTO TRA OSI E TCP/IP

Le architetture a strati del modello OSI e di Internet sono rappresentate nella figura 6. Esistono infiniti commenti su tema del confronto tra le due; quasi tutti si arrampicano sui vetri cercando impossibili paragoni tra protocolli, strati e quant'altro.
Vale qui sottolineare l'unica vera differenza, concettuale e non strutturale. OSI nasce come un modello di riferimento, TCP/IP nasce invece come pila di protocolli. OSI è il risultato di uno sforzo di modellizzazione per creare regole di comportamento nell'interconnessione tra sistemi aperti. Proprio per questo OSI è più importante per quello che *non* dice rispetto a quello che dice (OSI infatti non dice nulla circa la dimensione dei sistemi, la loro dislocazione geografica, la qualità della rete di trasporto, sul numero di macchine in cui le varie funzioni vengono realizzate ecc.). Nell'OSI la definizione dei protocolli è stata aggiunta in un secondo tempo e rappresenta la componente meno significativa: i protocolli OSI sono morti prima di nascere, uccisi dalla loro generalità e universalità (che fa rima con ambiguità e complessità). TCP/IP è invece il risultato di uno sforzo per realizzare un sistema di scambio dati semplice, robusto, di facile e sicura interpretazione, implementabile in modo *quick and dirty*. Del resto il motto dell'IETF, l'ente che presiede allo sviluppo delle norme di riferimento per Internet, è *We believe in rough consensus and running code*.
Un altro punto importante è tenere presente che la razionalizzazione e astrazione definita con il modello OSI è storicamente posteriore alla concezione delle reti che oggi conosciamo come Internet; inoltre il punto di vista preso da coloro che hanno sviluppato quest'ultimo paradigma è assumere per dato di fatto l'eterogeneità tecnologica delle reti locali e geografiche usate per interconnettere i sistemi di elaborazione (le co-

[2] Nel caso di schede WiFi l'ascolto (*carrier sensing*) costa circa l'80% del consumo energetico richiesto in trasmissione.

siddette sotto-reti: reti SNA, DecNet, Appletalk, LAN Ethernet, reti wireless) e puntare a definire un livello di inter-rete per realizzare l'interconnessione globale (il collante universale è IP) e fornire il supporto per l'esecuzione della più ampia varietà di processi applicativi. Illustra efficacemente questa concezione dell'architettura Internet il cosiddetto modello a clessidra, dovuto a Henning Schulzrinne (Figura 7).

Nell'architettura Internet si distingue quindi un livello di sotto-rete, che comprende gli strati bassi dell'architettura: può ridursi agli strati fisico e di collegamento (come nel caso delle LAN e dell'ATM) può essere incredibilmente complesso (si veda il caso delle reti dorsali del GPRS).

Sopra questo macro-livello basso si pone il livello unificante di inter-rete, la cui funzione specifica è indirizzamento, multiplazione e commutazione. Queste funzioni sono svolte da un unico protocollo, l'*Internet Protocol* (IP), corredato di alcuni altri protocolli ausiliari (ICMP, ARP, RARP).

Il primo strato con significato da estremo a estremo è quello sopra al livello IP, corrispondente grosso modo al livello di trasporto dell'architettura OSI, così come il livello IP corrisponde essenzialmente allo strato di rete. I due principali protocolli di questo livello sono il *Transmission Control Protocol* (TCP) e lo *User Datagram Protocol* (UDP).

Infine, al di sopra del livello di trasporto si colloca il livello applicativo, corrispondente praticamente agli strati di utilizzazione del modello OSI, cioè quelli di sessione, presentazione e applicativo vero e proprio.

Da questa pur sommaria descrizione si nota il blando grado di dettaglio con il quale nel modello Internet si descrivono i livelli bassi dell'architettura, rispetto all'attenzione in essi posta dal modello OSI e la prolissa e industrialmente poco prolifica visione del modello OSI relativamente agli strati alti, compattati dal modello Internet in un generico livello applicativo, la cui effettiva realizzazione è tutta demandata al software applicativo residente sui sistemi terminali e in eventuali sistemi intermedi (*proxy, media gateway* ecc.). È chiaro che i due modelli rispondono a esigenze e visioni maturate in mondi diversi, almeno nel momento in cui i

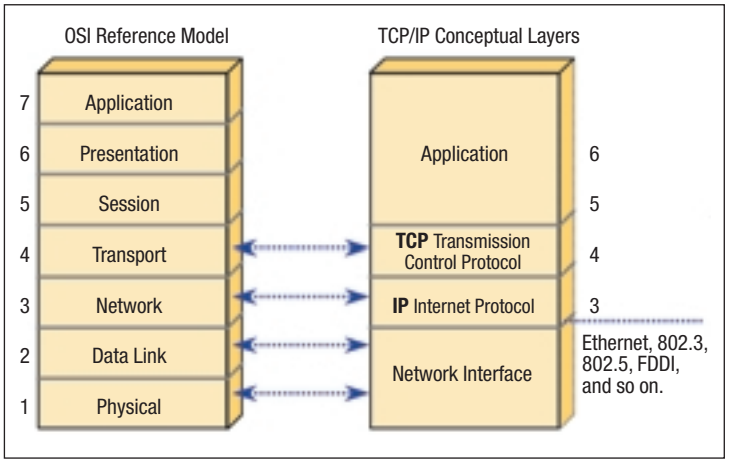

FIGURA 6
Pile protocollari OSI e Internet a confronto

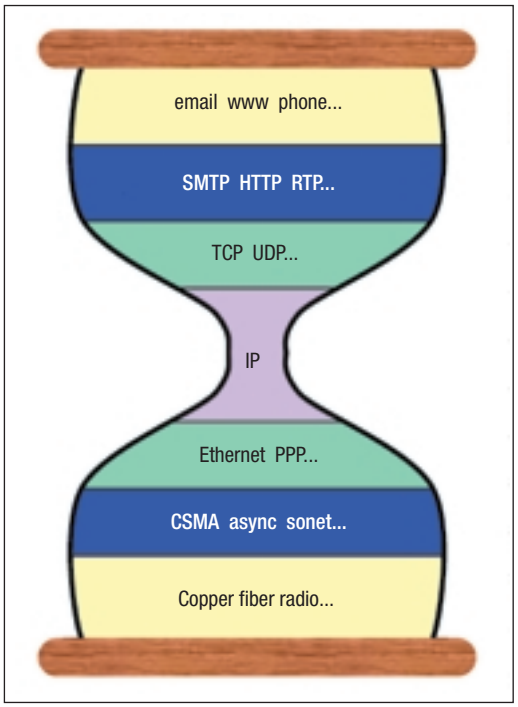

FIGURA 7
Modello a clessidra dell'architettura di Internet

modelli si affermarono (anni '70 e primissimi anni '80): l'industria dei computer e le comunicazioni di dati per il modello Internet; il mondo delle telecomunicazioni classiche, telefoniche prima di tutto, per il caso del modello OSI. Traccia di questo si ritrova, tra gli altri aspetti, nella matrice dei principali Enti di normativa promotori dello sviluppo di nuovi protocolli: ITU-T, ETSI, 3GPP per le reti cellulari, le reti di trasporto e in generale tutte quelle con un forte contenuto di tecnologia di telecomunicazione (vicine ai livelli più bassi dell'architettura: fisico,

di collegamento); IETF, IEEE vari Fora e consorzi industriali per quanto riguarda Internet, le reti wireless, le reti in area locale, in generale quelle reti miranti in primo luogo all'interconnessione dei computer (vicine per lo più ai livelli dalla rete in su, quasi esclusivamente realizzate in software, al più firmware per il caso delle reti wireless e delle schede per LAN).
La storia recente ha visto il crescente successo del modello Internet. Questo spostamento di paradigma avvenuto a partire dalla metà degli anni '90, manifestato anche nella terminologia (si parla oggi di ICT, non di telecomunicazioni), tra le molte conseguenze ha avuto anche un significativo impatto sul modo di concepire, ideare, sviluppare e gestire i protocolli. Da un approccio "telecom" connotato da tecnologie ad hoc, spesso dipendenti da specifici sviluppi hardware, comunque con piattaforme di sviluppo che solo grossi laboratori aziendali specializzati potevano affrontare (esempio classico: ISDN), si è passati con il prevalere del modello Internet a protocolli realizzati soprattutto in software, sia nel *kernel* dei sistemi operativi sia spesso con programmi che girano in *user space*; sempre più spesso i protocolli sono materialmente realizzati usando codice sorgente in C, java o altri linguaggi di livello relativamente alto; sempre più spesso c'è un intreccio inestricabile tra il software dei computer e quello dei protocolli rivolti alla soluzione di problemi di comunicazione e spesso il tutto è fornito come codice open source, soprattutto con il progressivo affermarsi di Unix nei suoi vari dialetti (vedi Linux sui PC commerciali); anche i protocolli dei livelli di collegamento del MAC e persino molte funzioni tradizionalmente appartenenti al livello fisico, sono oggi realizzati utilizzando DSP o FPGA, con uno sforzo di sviluppo focalizzato sulla programmazione a livello relativamente alto, grazie agli strumenti di CAD e compilazione che fungono da intermediari tra il codice sorgente prodotto dallo sviluppatore e quello che deve essere caricato nell'hardware. Queste tendenze hanno abbassato drammaticamente la soglia economica per lo sviluppo, la sperimentazione, la valutazione e misura di protocolli di comunicazione, nello stesso tempo fondendo le competenze necessarie con quelle proprie della computer industry. Se si vuole anche essere polemici, queste tendenze hanno anche ingenerato qualche confusione e concesso spazio a tecnici di scarsa professionalità.

5. CROSS-LAYERING OVVERO IL MODELLO OSI ULTIMA PAROLA?

Le architetture stratificate poggiano su un assioma: le funzioni attivate per la cooperazione di processi applicativi attraverso una rete di comunicazione sono raggruppate in insiemi omogenei (gli strati) e i protocolli che realizzano queste funzioni sono definiti all'interno di ciascuno strato *indipendentemente* dagli altri se non per i vincoli imposti dalle interfacce tra strati adiacenti (le specifiche delle primitive di servizio tra strato utente e strato fornitore); in altri termini, il servizio che le entità di uno strato si aspettano da quelle dello strato inferiore è specificato a livello dell'interfaccia tra i due strati (il Service Access Point tra lo strato $N+1$ e N, (N)-SAP, vedi Figura 5), ma non è specificato il *modo* in cui le entità dello strato N devono poi realizzare quel servizio (i protocolli dello strato N).
È un tipo di approccio che fa tesoro della massima "divide et impera", ed è un classico esempio di decomposizione di quello che può essere concepito come un grande problema di ottimizzazione vincolata in sottoproblemi collegati gerarchicamente, che possono essere tuttavia risolti separatamente, raggiungendo tipicamente un sub-ottimo come soluzione globale (ma alla soluzione si arriva!). Questo è l'approccio che ha reso possibile la definizione delle reti cellulari, delle reti di trasporto (SDH, ottiche), della stessa rete Internet.
Esistono alcuni contesti dove l'ottimizzazione perseguibile derogando alla rigida regola di separazione dei protocolli di strati diversi è abbastanza pagante da superare i vantaggi di flessibilità e potenziale semplicità offerti dalla decomposizione in strati internamente indipendenti. In questi casi, gli algoritmi che sono svolti dalle entità che partecipano al protocollo utilizzano informazioni (eventi, misure, dati) appartenenti anche ad altri strati per ottimizzare le prestazioni ottenibili: si parla di

protocolli *cross-layer*. È questo un approccio che ha riscontrato una fortuna crescente negli ultimi anni in almeno due contesti molto significativi: le reti wireless e il controllo della congestione nelle reti a pacchetto.
Nel primo caso, quando si tratta di servire una popolazione di utenti che condividono una risorsa radio, esiste una relazione stretta tra gli effetti di scelte effettuate nei protocolli di livello fisico, MAC, di collegamento e di rete. Per esempio, si pensi alle operazioni di scheduling dei pacchetti effettuati da una stazione base, utilizzando come metriche guida la QoS differenziata da offrire ai diversi flussi di dati oppure criteri di *fairness*; si tratta di una funzione collocata tipicamente nello strato di collegamento o di rete. È chiaro che questa funzione ha un nesso molto stretto con la gestione delle risorse radio in termini di quali codici, potenze, frequenze, time slot usare in trasmissione, la codifica e modulazione dei dati, tutte decisioni proprie dei protocolli di strato fisico. Molti lavori sia teorici sia sperimentali hanno mostrato gli ampi margini di guadagno possibili se si formula un problema congiunto (cross layer appunto) che tenga congiuntamente conto delle caratteristiche del *traffico* e del *canale radio* [7, 8].
In alcuni contesti, come le reti di sensori, il *cross-layering* porta a definire problemi che coinvolgono congiuntamente anche più di due strati. I parametri da usare per trasmettere i dati (potenza, modulazione, codifica), quando trasmettere tenendo conto di obiettivi di portata (*throughput*) equità di accesso e contenimento dell'interferenza o al limite delle collisioni, l'instradamento da seguire secondo un paradigma di tipo multihop (raggiungere la destinazione con un unico passo, consumando maggiore potenza e provocando più interferenza oppure effettuare più passi, aggiungendo quindi ritardo, consumo energetico e risorse di calcolo e memoria dei nodi intermedi?) sono tutti aspetti che devono essere considerati assieme se si vuole raggiungere un'efficienza ottima complessiva del sistema in termini di prestazioni e costo. Nel caso delle reti di sensori, il problema è abbastanza delimitato e le condizioni operative sufficientemente "estreme" da motivare un approccio olistico, di tipo *cross-layer*, anziché lo sviluppo di protocolli indipendenti nel senso delle architetture stratificate.
Altro esempio di *cross-layering* è dato dalle funzioni di controllo di congestione, svolta nei protocolli di trasporto da estremo a estremo (per esempio mediante controlli a finestra variabile e adattiva, come nel TCP) e le politiche di scheduling e gestione delle code interne ai nodi della rete, le cosiddette politiche di *Active Queue Management*, AQM. Queste ultime sono funzioni ovviamente appartenenti a protocolli dello strato di rete, ma il loro effetto sulle decisioni e le prestazioni ottenibili dai protocolli di trasporto sono tali che un approccio di ottimizzazione congiunta *cross layer* è posto all'attenzione della comunità di ricercatori e sviluppatori che lavorano su questi problemi [9].
Come tutti i modelli realmente utili, anche il modello OSI e gli analoghi modelli alternativi introducono astrazioni e concetti di base, fondamentali per uno sviluppo ordinato ed efficace di sistemi complessi, ma vanno capiti nelle loro motivazioni e usati con capacità di adattamento ai contesti tecnologici e applicativi specifici, senza dogmatismo. Del resto anche l'indipendenza funzionale e quindi realizzativa dei protocolli di strati distinti, infranta dall'ottimizzazione *cross-layer*, ha dimostrato di conseguire sì sub-ottimi, ma di potersi avvalere con estrema tempestività delle diverse velocità di evoluzione delle tecnologie base dei diversi strati: si veda l'esempio del Wi-Fi, nel quale il livello MAC è rimasto invariato nella sostanza dal 1997 a oggi[3], mentre sono state messe a punto quattro principali versioni del livello fisico (IEEE 802.11 base e successivamente IEEE 802.11b/a/g).

6. L'EVOLUZIONE DELLO STRATO APPLICATIVO

L'evoluzione del modello OSI é insita nella sua stessa struttura; OSI é assolutamente generale. La generalità del modello OSI e del suo rivoluzionario concetto di strato permette infatti di cambiare, sostituire, arricchire di funzioni uno "strato" senza alterare l'impalcatura.

[3] Solo con gli imminenti standard IEEE 802.11e e 802.11n si intende rimettere mano alla definizione del protocollo MAC.

Le nuovi reti wireless (WiFi, WiMax) possono rimpiazzare le tecnologie esistenti dando migliori o differenti servizi di connettività ai DTE finali e lasciandone inalterate le funzionalità applicative. Se si hanno reti di trasporto a larga banda o a qualità garantita, si possono costruire sessioni multimediali in cui sono presenti voce ed immagini. I principi non cambiano.

A livello applicativo abbiamo avuto le evoluzioni più significative – non in termini concettuali, ma in termini di completezza funzionale. Lo strato applicativo ha infatti tratto enormi benefici dal Web, ambiento aperto e distribuito. Ai tempi in cui l'OSI è stato concepito, gli unici servizi applicativi ipotizzabili erano quelli associati al trasporto dei dati (*file transfer*), alla consegna dei messaggi (posta elettronica) oppure alla cooperazione transazionale per eseguire funzioni distribuite ma necessariamente sincrone (CCR, *Commitment and Concurrency Recovery*, utilizzato per transazioni a più livelli o aggiornamenti multipli di basi di dati); in sostanza erano i servizi ipotizzabili in architetture costituite essenzialmente da mainframe ed elaboratori satellite.

Il Web, ambiente naturale in cui eseguire applicazioni con presentazione e metodi di accesso standard, permette di arricchire enormemente l'insieme dei servizi applicativi disponibili. Il connubio tra Web ed Internet, lo sviluppo di infrastrutture fisiche in cui le server farm sono sempre più un'appendice naturale della rete di telecomunicazione, permettono di creare un midlleware applicativo in cui le applicazioni sono una collezione di servizi, invocati ed eseguiti sulla base delle necessità, senza nulla imporre alla loro reale dislocazione fisica.

Nasce dunque l'idea dei *Web Services*, concettualmente analoghi agli *Application Service Elements* del modello OSI, ma molto più universali.

Il *Web Service* è un *middleware* che permette di trasformare un'applicazione (un programma che meccanizza un processo di business e produce dei risultati) in un "oggetto", dotato di attributi e metodi, utilizzabile come un mattoncino del lego in un ambiente distribuito ed aperto. Per far questo e' necessario disporre di una funzione di descrizione del servizio, di una sua catalogazione e di un meccanismo per poterlo invocare, passargli degli input e ricevere dei risultati. L'insieme di un'architettura aperta ed universale come il TCP/IP e di applicazioni create con la logica dei *Web Services* rappresenta la realizzazione più significativa del modello concepito dalla ISO nei lontani anni '80. Se poi pensiamo di aggiungere a questi servizi anche un'autostrada intelligente che coordini il flusso di informazioni scambiate tra i vari moduli applicativi (con funzioni di gestione delle code, delle priorità, delle *retry* ecc.) - ossia di aggiungere ciò che è comunemente chiamato *enterprise bus* - abbiamo creato una nuova impostazione dei sistemi informativi in cui l'enfasi si sposta dalla gerarchia dell'elaborazione alla granularità delle funzioni applicative e alla loro utilizzabilità nell'ambito delle applicazioni finali, sempre più governate dalla dinamica del business (SOA, *Service Oriented Architecture*).

Scendendo un po' nei dettagli, i *Web Service* definiti sono:

- WSDL (*Web Service Definition Language*): descrive cosa fa un *Web Service* nonché la sua operatività (come può essere invocato, come gli si possono passare i parametri);
- UDDI (*Universal Discovery Description and Integration*): facilita la pubblicazione e la ricerca dei servizi disponibili. Sono disponibili funzioni di "pagine bianche" e "pagine gialle" (classificazioni per nome specifico o categoria);
- SOAP (*Service Oriented Application Protocol o Simple Object Application Protocol*): offre – sulla base dello standard XML- meccanismi standard per l'interscambio di informazioni.

Non è obiettivo di questo articolo descrivere in dettaglio la logica dei *Web Services* (Mondo Digitale ha già trattato questo argomento in modo completo ed autorevole – si veda a tale proposito "Un'introduzione ragionata al mondo dei *Web Services*", marzo 2004), ma evidenziare il loro posizionamento nell'ambito del "valore aggiunto" ai protocolli di comunicazione.

Abbiamo sostituito mainframe, calcolatori satellite, terminali con PC e applicazioni in esecuzione presso *Web Farms*. Abbiamo

cambiato il terminale 3270 con una pagina Web, abbiamo cambiato le l*ogical unit* di IBM con l'HTTP, abbiamo introdotto l'IP come busta universale per il trasporto e l'instradamento. Abbiamo insomma ottenuto ambienti più flessibili, ma viviamo ancora dei concetti introdotti dalla *Service Definition* del modello OSI. L'ISO 7498 ha ancora molto da insegnare a tutti, vent'anni dopo.

Bibliografia

[1] Tanenbaum A.S.: *Computer Networks*. 4-th edition, Prentice Hall, 2003.

[2] IBM, Systems Network Architecture, 1978.

[3] Listanti M., et alii: *Telematica*. Capitolo del testo "Manuale di Informatica", Calderini, 1986, p. 781-902.

[4] Elementi di procedura, descrizione della trama HDLC e della modalita' "balanced" e "normal".

[5] Kleinrock L.: *Queueing Systems*. Computer applications, Vol. II, Wiley, 1976.

[6] Modello OSI, Service Definition e Protocol Specification.

[7] Song G., Li Y.: Cross-Layer Optimization for OFDM Wireless Networks - Part I: Theoretical Framework. *IEEE Transactions on Wireless Communications*, Vol. 4, n. 2, March 2005, p. 614- 624.

[8] Lau V.K.N., Kwok Y.-K.R.: *Channel adaptive technologies and cross layer designs for wireless systems with multiple antennas*. J. Wiley, 2006.

[9] Wang Jiantao, Li Lun, Low Steven H., Doyle John C.: Cross-Layer Optimization in TCP/IP networks. *IEEE/ACM Trans. on Networking*, Vol. 13, n. 3, Giugno 2005, p. 582-568.

GIACOMO ZANOTTI laureato in Ingegneria elettronica al Politecnico di Milano nel 1973, ha seguito dal nascere il tema dell'Information Communication Technology. Ha ricoperto incarichi di direttore marketing e di responsabile di Business Unit di system integration in primarie aziende di informatica e telecomunicazioni. Attualmente collabora in ITnet (Internet Service Provider del Gruppo Wind) alla definizione delle strategie di sviluppo business e allo sviluppo dei canali di vendita. Ha interesse all'ambiente accademico e tecnico-scientifico, che sviluppa come membro del consiglio AICA e con attività di docenza in Master organizzati da Università (Milano, Padova). Ha ricoperto per quattro anni il ruolo di membro italiano nella commissione Europea IST-prize.
E-mail : giacomo.zanotti@gmail.com

ANDREA BAIOCCHI si è laureato in Ingegneria Elettronica nel 1987 e ha conseguito il Dottorato di Ricerca in "ingegneria dell'Informazione e della Comunicazione" nel 1992 presso l'Università di Roma "La Sapienza". Dal gennaio 2005 è Professore Straordinario nel settore Telecomunicazioni presso la Facoltà di Ingegneria dell'Università "La Sapienza".
I principali contributi scientifici di Andrea Baiocchi sono sui modelli e algoritmi per il controllo del traffico in reti ATM e TCP/IP, sulla teoria delle code, sulla gestione delle risorse radio in reti cellulari. I suoi attuali interessi di ricerca sono concentrati sui modelli per l'analisi e il dimensionamento di reti a pacchetto con controllo della congestione da estremo a estremo, secondo il paradigma del TCP, e sul "mobile computing," in particolare adattamento del TCP al canale wireless e sulle gestione cross-layer delle risorse nell'interfaccia radio. Queste attività sono state e sono tutt'ora scolte nell'ambito di progetti sia nazionali (CNR, MIUR) sia internazionali (UE, ESA), ricoprendo anche posizioni di responsabile di unità di ricerca. Andrea Baiocchi ha al suo attivo oltre ottanta pubblicazioni su riviste scientifiche e su atti di conferenze internazionali, ha preso parte alle Commissioni Tecniche di Programma di sedici convegni internazionali, tutti nel settore delle reti di tlc; ha anche fatto parte della redazione del Notiziario Tecnico di Telecom Italia per dieci anni.
E-mail: andrea.baiocchi@uniroma1.it