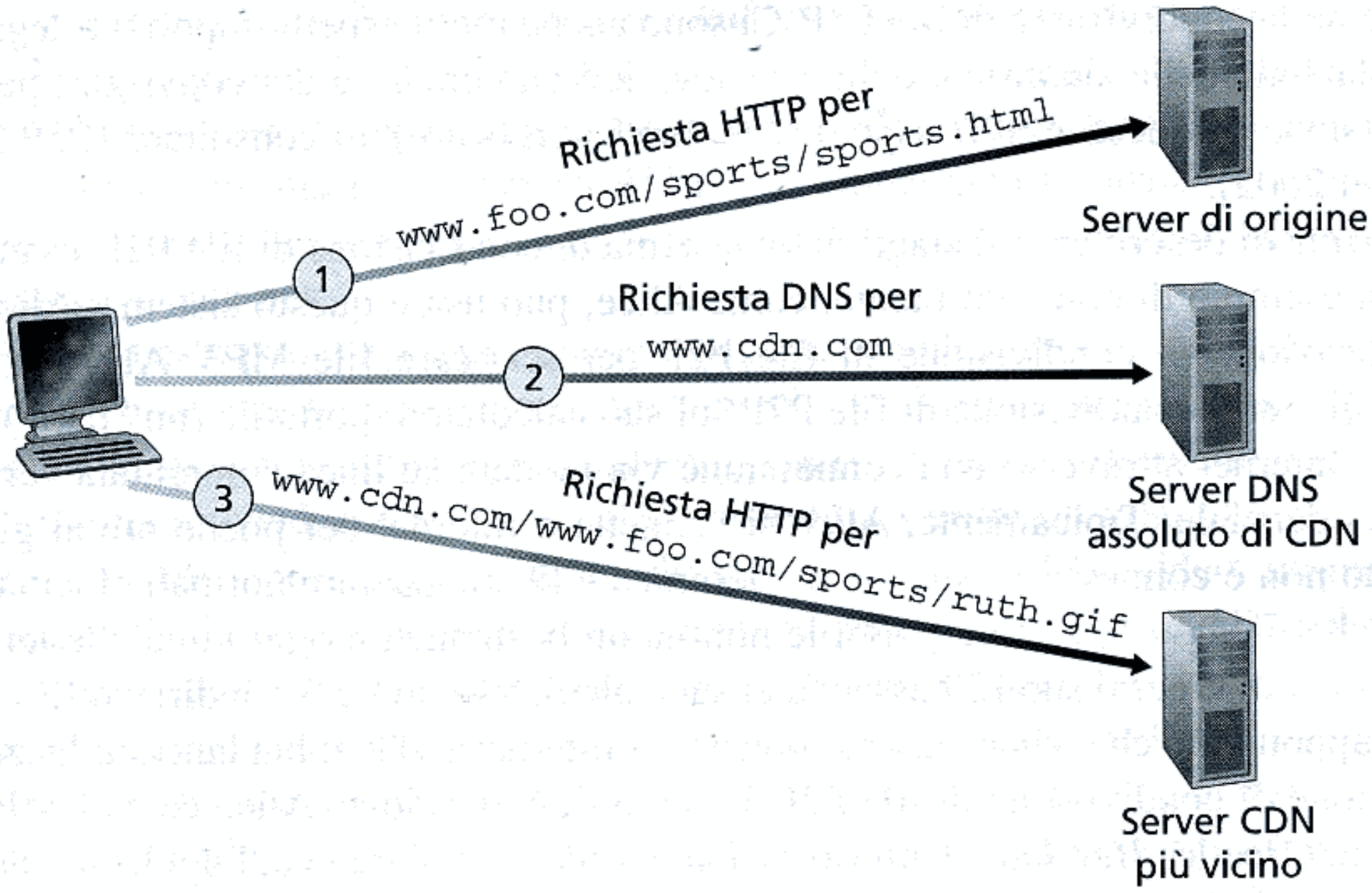

Figura 2.31 ♦ Le CDN usano il DNS per indirizzare le richieste al server CDN più vicino.

Quello che rimane da spiegare è come fa un'azienda di CDN a determinare il miglior server CDN per l'host richiedente. Sebbene ogni azienda di CDN abbia il suo modo proprietario di fare questo, non è difficile farsi un'idea approssimativa di come si procede. Per ogni ISP di accesso a Internet (che contiene potenziali host richiedenti), l'azienda di CDN tiene traccia del miglior server CDN per quell'ISP. L'azienda di CDN determina il miglior server CDN sulla base della sua conoscenza delle tabelle di instradamento di Internet (in particolare, le tabelle BGP, che discuteremo nel Capitolo 4), delle stime del tempo di round-trip e di altre misure che possiede dai suoi diversi server verso varie reti di accesso; per una trattazione si può vedere [Verma 2001]. In questo modo, la CDN stima quale server CDN fornisce il minor tempo di risposta all'ISP. La CDN esegue questa operazione per un gran numero di ISP di accesso in Internet e usa queste informazioni per configurare il server DNS responsabile.

Abbiamo discusso le CDN prevalentemente nel contesto della distribuzione di contenuti Web. Le CDN sono molto usate anche per la distribuzione di flussi audio e video. Per distribuire contenuti di tipo audio e video, i server CDN spesso supportano protocolli di controllo dei flussi come RTSP (vedi il Capitolo 6).

2.9.3 Condivisione di file da pari a pari

Abbiamo appena imparato che un utente può scaricare oggetti da server Web di origine che sono gestiti da fornitori di contenuti, da cache proxy che sono gestite da ISP, o da server CDN che sono gestiti da aziende di CDN. Ma c'è ancora un'altra possibilità. I PC all'accesso della rete (chiamati "pari", *peer*) possono scaricare oggetti direttamente l'uno dall'altro! Questa semplice osservazione è l'essenza dei paradigmi di condivisione di file da pari a pari (*peer-to-peer*, P2P), compresi Napster [Napster 2002], Gnutella [Gnutella 2002], Freenet [Freenet 2002] e FastTrack [FastTrack 2002]. La nostra discussione che segue sarà centrata sugli aspetti di comunicazione e

di rete nella condivisione di file P2P. Ci sono anche molti aspetti importanti legati alla violazione della sicurezza, della privacy, dell'anonimato e del copyright; per una discussione di questi e altri aspetti il lettore interessato può consultare [P2P 2001; Freenet 2002].

Prima di descrivere i dettagli di un sistema di condivisione di file P2P, esaminiamo un esempio di come un utente, come Alice, può usare questo sistema. Alice usa l'applicazione di condivisione di file P2P per scaricare file MP3. Alice avvia il software per la condivisione di file P2P sul suo calcolatore portatile (un "pari") e accede a Internet attraverso una connessione via modem su linea commutata verso un ISP residenziale. Tipicamente, Alice si connette a Internet per poche ore al giorno; quando non è connessa, la sua linea telefonica è libera per fare normali chiamate telefoniche. Il suo calcolatore portatile non ha un hostname, e ogni volta che lei si riconnette a Internet il suo ISP assegna al suo calcolatore un nuovo indirizzo IP.

Supponiamo che Alice sia ora connessa a Internet e che abbia lanciato la sua applicazione di condivisione di file P2P. Usando l'applicazione, Alice cerca l'MP3 del brano dei Beatles *Hey Jude*. Dopo aver dato il comando di ricerca, l'applicazione mostra una lista di pari, che sono tutti attualmente connessi a Internet e hanno una copia di *Hey Jude* da condividere. Ciascuno di questi pari è tipicamente un normale PC, di proprietà di un normale utente di Internet come Alice. Per ogni pari della lista, l'applicazione può mostrare alcune informazioni ulteriori, come la larghezza di banda di accesso del pari e la stima del tempo necessario per scaricare il file. Alice quindi richiede il file MP3 da uno dei pari, diciamo il PC di Bob. Viene stabilita una connessione diretta TCP tra il PC di Alice e il PC di Bob, e il file MP3 viene inviato dal PC di Bob al PC di Alice. Se Bob disconnette inavvertitamente il suo PC da Internet mentre Alice sta scaricando, allora il software di condivisione di file P2P può cercare di ottenere il resto del file da un altro pari che lo possiede. Inoltre, mentre Alice sta scaricando *Hey Jude* da Bob, un altro pari, diciamo il PC di Claire, può simultaneamente scaricare un altro file MP3, diciamo il pezzo dei Rolling Stones *Angie*, direttamente dal PC di Alice. Quindi, ogni pari partecipante è sia un distributore sia un consumatore di contenuti.

La condivisione di file P2P è un paradigma di condivisione di contenuti avvincente, in quanto tutti i contenuti vengono trasferiti direttamente tra normali pari, senza passare attraverso server di terza parte (come il server Web di origine, le cache del Web o i server CDN). Quindi, il P2P sfrutta le risorse (banda, memoria e CPU) di una molteplicità di pari (a volte milioni!) per distribuire i contenuti. In altre parole, la condivisione di file P2P è ampiamente scalabile.

Sebbene nessun server centralizzato di terza parte partecipi al trasferimento di file, è importante ricordare che la condivisione di file da pari a pari usa comunque il paradigma client-server che è ampiamente prevalente in Internet. In effetti, il pari richiedente è il client, e il pari scelto è il server. Il file viene inviato dal pari server al pari client con un protocollo di trasferimento di file. Dal momento che qualunque pari può richiedere o essere scelto, tutti i pari devono essere in grado di azionare sia il lato client sia il lato server del protocollo di trasferimento di file. Ora consideriamo il caso in cui il protocollo di trasferimento di file P2P sia HTTP, che è il caso più tipico. Continuando con il nostro esempio precedente, quando Alice sceglie Bob per scaricare *Hey Jude*, il PC di Alice manda a Bob una richiesta HTTP per *Hey Jude*, e Bob

manda una risposta HTTP contenente *Hey Jude*. È da notare che mentre sul calcolatore di Alice sta girando l'applicazione di condivisione di file P2P, il suo pari è sia un client Web sia un **server Web temporaneo**. Il suo pari è un server Web in quanto fornisce contenuti all'interno di risposte HTTP; è temporaneo in quanto è connesso a Internet in modo intermittente, e può avere un nuovo indirizzo IP ogni volta che viene connesso a Internet.

Fino a questo punto abbiamo spiegato la parte semplice della condivisione di file P2P, cioè, come i contenuti vengono direttamente trasferiti da un pari a un altro. Ma non abbiamo spiegato come un pari determina quali pari hanno il contenuto richiesto. In un sistema di condivisione di file P2P esiste tipicamente un gran numero di pari connessi, e ogni pari ha oggetti da condividere, compresi MP3, video, immagini e programmi. Se il pari X è interessato a ottenere un particolare oggetto, allora il pari X deve avere un modo di determinare gli indirizzi IP dei pari connessi che hanno copie dell'oggetto desiderato. Poiché i pari si connettono e si disconnettono, questo è un problema non banale. In seguito discuteremo tre architetture per localizzare i contenuti, ciascuna delle quali è stata usata da diversi sistemi di condivisione di file P2P. Il lettore interessato è anche invitato a vedere i lavori di ricerca [Chord 2001; Pastry 2001; CAN 2001] che forniscono soluzioni innovative per il problema della localizzazione dei contenuti.

Directory centralizzata

Uno degli approcci più immediati per localizzare i contenuti è fornire una **directory centralizzata**, come nel caso di Napster, che è stata la prima azienda commerciale a fornire un'applicazione per la distribuzione di MP3 da pari a pari su larga scala. In questa architettura il servizio di condivisione di file P2P usa un grande server (o una server farm) per fornire il servizio di directory. Come è mostrato nella Figura 2.32, quando un utente lancia l'applicazione di condivisione di file P2P, l'applicazione

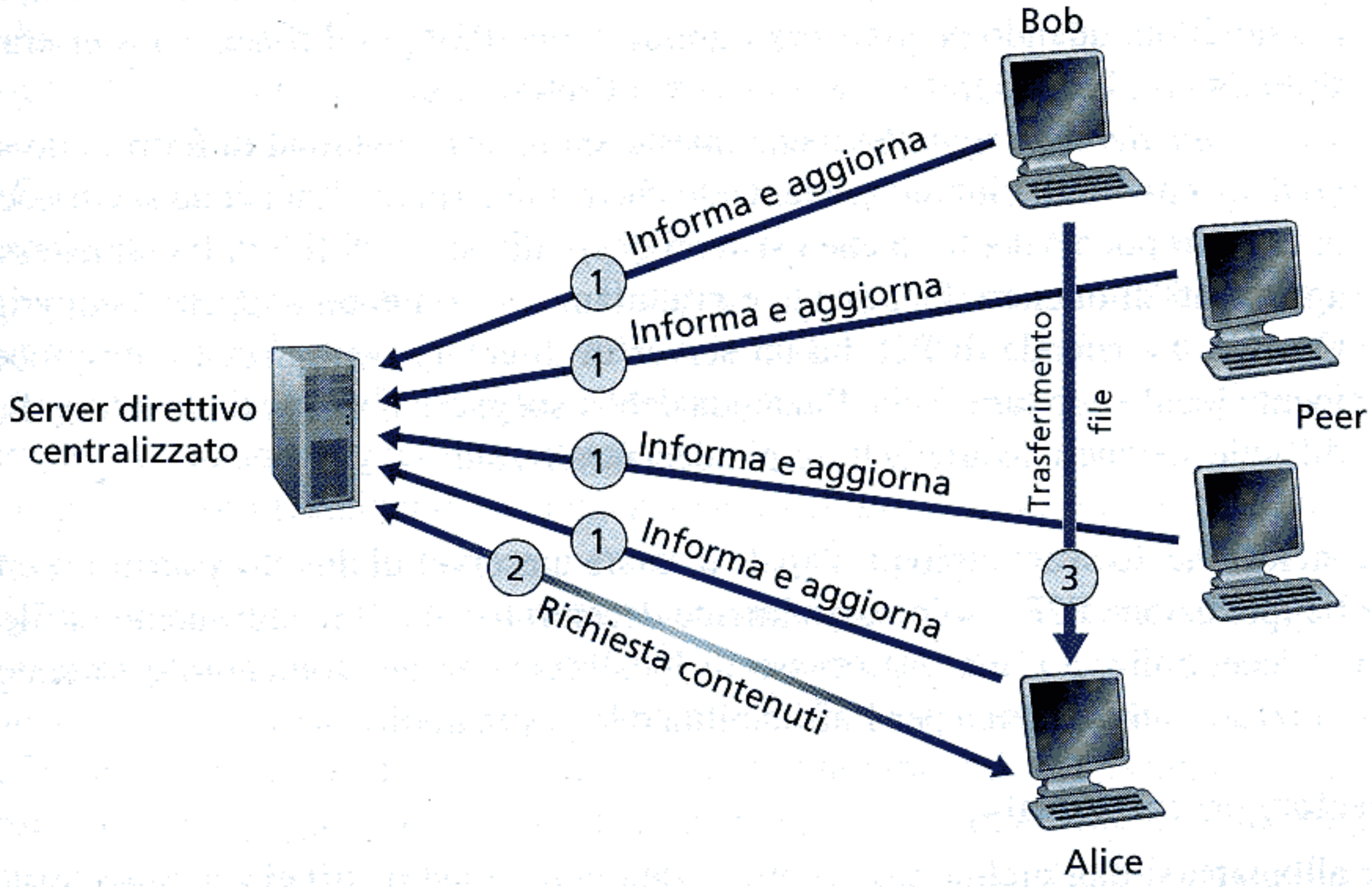

Figura 2.32 ♦ Il paradigma peer-to-peer con directory centralizzata.

contatta il server di directory. In particolare, l'applicazione che gira sul pari informa il server di directory del suo indirizzo IP e dei nomi degli oggetti contenuti nel suo disco locale che rende disponibili per la condivisione (per esempio, i titoli di tutti gli MP3 memorizzati). In questo modo, il server di directory sa quali oggetti il pari ha a disposizione per la condivisione. Il server di directory raccoglie queste informazioni da ogni pari che si attiva, creando così un database centralizzato, dinamico che mette in corrispondenza ogni nome di oggetto con una serie di indirizzi IP. Quando un pari attivo ottiene un nuovo oggetto, o rimuove un oggetto, informa il server di directory, in modo che il server di directory possa aggiornare il suo database.

Per mantenere aggiornato il suo database il server di directory deve poter determinare quando un pari si disconnette. Un pari può disconnettersi chiudendo la sua applicazione client P2P o semplicemente disconnettendosi da Internet. Un modo di mantenere traccia di quali pari rimangono connessi è di mandare periodicamente dei messaggi ai pari per vedere se rispondono. Un altro modo a disposizione del server di tenere traccia dei pari connessi è di mantenere una connessione TCP permanente con ogni pari connesso; se la connessione TCP termina, il pari è disconnesso. Se il server di directory accerta che un pari non è più connesso, rimuove l'indirizzo IP del pari dal database.

Usare una directory centralizzata per localizzare i contenuti è concettualmente immediato, ma ha una serie di svantaggi:

- *Unico punto di rottura.* Se il server di directory si blocca, allora si blocca l'intera applicazione P2P. Anche se si usa un gruppo di server (*server farm*) con server ridondanti, possono interrompersi le connessioni Internet alla server farm, con conseguente blocco dell'intera applicazione.
- *Collo di bottiglia delle prestazioni.* In un grande sistema P2P, con centinaia di migliaia di utenti connessi, un server centralizzato deve mantenere aggiornato un enorme database e deve rispondere a migliaia di richieste al secondo. Infatti, nell'anno 2000, quando Napster era l'applicazione P2P più diffusa, Napster era afflitto da problemi di traffico al suo server centralizzato.
- *Violazione del copyright.* Sebbene questo sia un argomento al di fuori dello scopo di questo libro, citiamo brevemente che l'industria dell'incisione si è preoccupata (a dir poco!) del fatto che i sistemi di condivisione di file P2P permettessero agli utenti di ottenere facilmente e gratuitamente contenuti soggetti a copyright. Quando un'azienda di P2P ha un server di directory centralizzato, un procedimento legale può fare sì che l'azienda debba spegnere il server di directory. È più difficile spegnere le architetture più decentralizzate.

In conclusione, lo svantaggio principale di usare un server di directory centralizzato è che l'applicazione P2P è solo parzialmente decentralizzata. Il trasferimento di file tra pari è decentralizzato, ma il processo di localizzazione del contenuto è fortemente centralizzato: un problema per l'affidabilità e le prestazioni.

Directory decentralizzata

Per allontanarsi dall'architettura centralizzata, è naturale distribuire la directory per la localizzazione del contenuto tra i pari stessi. Il sistema di condivisione di file P2P

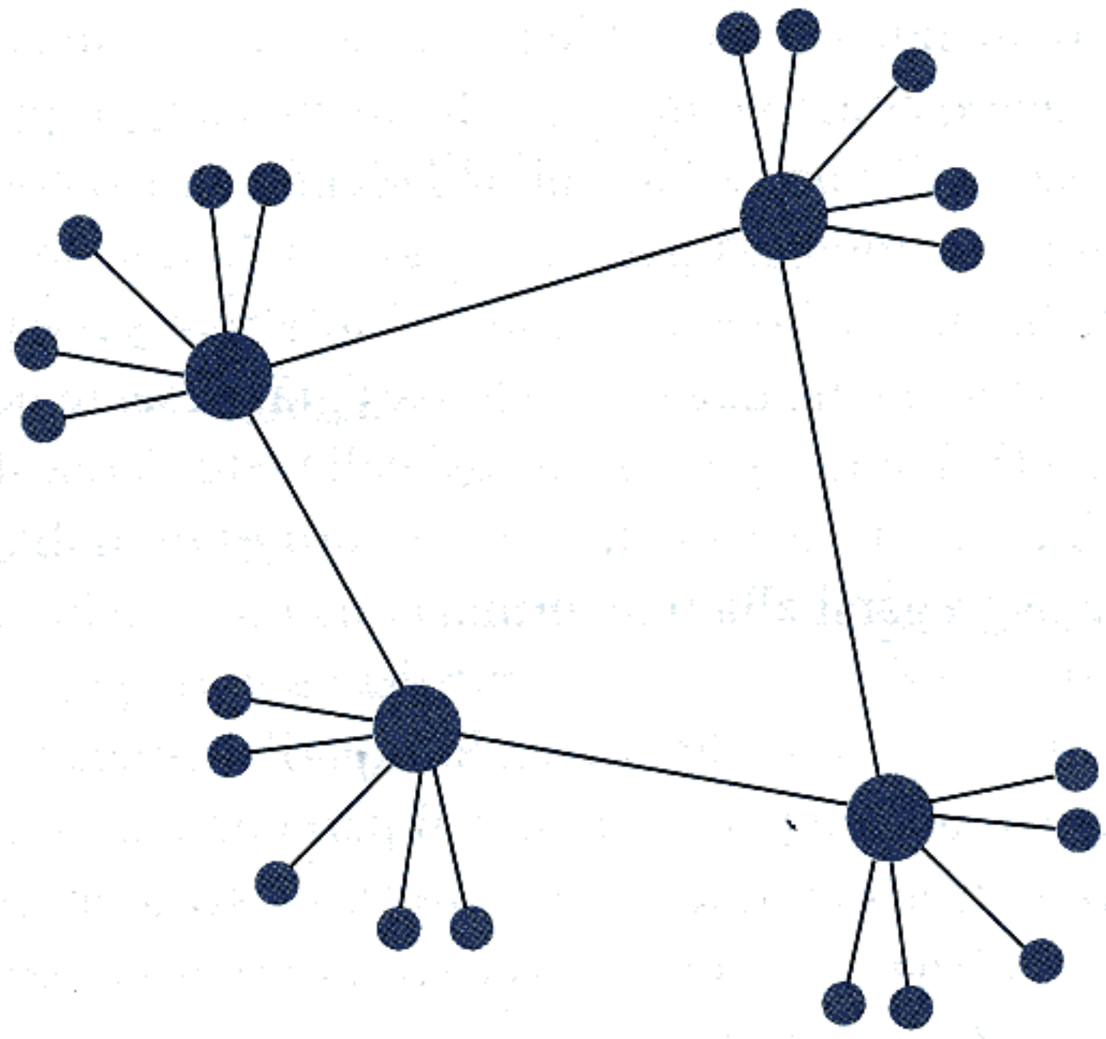

Figura 2.33 ♦ Rete gerarchica sovrapposta per la condivisione di file P2P con directory decentralizzata.

KaZaA/FastTrack [FastTrack 2002], ampiamente diffuso nel 2001-2002, ha adottato questo approccio.

Come è mostrato nella Figura 2.33, un certo numero di pari (decine, centinaia o addirittura migliaia) sono nominati capigruppo. Quando un pari si connette all'applicazione P2P, il pari viene assegnato a uno dei capigruppo. Dopo aver conosciuto l'indirizzo IP del suo capogruppo, il pari contatta il suo capo e lo informa dei contenuti che si appresta a condividere. Il capogruppo mantiene un database, che mette in corrispondenza i nomi dei contenuti con gli indirizzi IP, per tutti i pari che sono stati assegnati al suo gruppo. In questo modo, ogni gruppo diventa un minisistema di condivisione di file P2P, in cui il capogruppo ha un ruolo simile al server di directory nell'architettura centralizzata descritta sopra. È importante notare che il database di un capogruppo registra solo i contenuti all'interno del suo gruppo. Inoltre, un capogruppo non è un server dedicato; al contrario, è un normale pari.

Quando un pari, diciamo Alice, vuole localizzare un particolare oggetto, manda una richiesta al suo capogruppo. Il capogruppo risponde ad Alice con una lista di pari del suo gruppo che hanno l'oggetto. Il capogruppo può anche contattare altri capigruppo, e chiedere loro di mandare ad Alice la lista dei loro pari che hanno l'oggetto. In questo modo, ricevendo liste da molti capigruppo, Alice ottiene gli indirizzi IP di molti pari che hanno l'oggetto desiderato.

I pari e le loro relazioni di comunicazione formano una rete logica astratta, detta **rete sovrapposta** (*overlay network*), come mostrato nella Figura 2.33. In termini di teoria dei grafi, i nodi della rete sovrapposta sono i pari; c'è un ramo tra ogni pari e il suo capogruppo; e c'è un ramo tra ogni coppia di capigruppo che comunicano direttamente (si rilanciano richieste reciprocamente). È da notare che questi rami non sono link fisici di comunicazione, ma piuttosto link virtuali tra pari. Per esempio, anche se un pari e il suo capogruppo sono in ISP diversi in diversi continenti, essi sono direttamente connessi da un ramo nella rete sovrapposta, e sono quindi **vicini virtuali** nella rete sovrapposta.

A questo punto probabilmente vi domanderete come viene creata e mantenuta aggiornata la rete sovrapposta. In particolare, come fa un pari a diventare capogruppo, e in che modo un nuovo pari viene assegnato a un capogruppo? Poiché i pari si connettono e disconnettono in continuazione (compresi i pari capogruppo), la rete sovrapposta evolve dinamicamente. Affinché un pari possa aggregarsi alla rete sovrapposta, esso deve stabilire almeno un ramo tra sé e un qualche altro pari già presente nella rete; cioè, deve conoscere l'indirizzo IP di almeno un altro pari nella rete. A questo scopo, il sistema di condivisione di file P2P deve avere uno o più **nodi di bootstrap** (avviamento). Quando un pari vuole aggregarsi alla rete, prima contatta il nodo di bootstrap. Il nodo di bootstrap risponde con l'indirizzo IP di uno dei capigruppo, e quindi il pari crea un ramo verso quel capogruppo. Inoltre, quando il pari si collega inizialmente con il nodo di bootstrap, quest'ultimo può designare il pari come nuovo capogruppo. Se il pari è un capogruppo, deve conoscere gli indirizzi IP di alcuni (o tutti) gli altri capigruppo. Poiché i nodi di bootstrap sono server sempre accesi, i pari possono utilizzare il DNS per localizzarli.

Una delle caratteristiche attraenti di questo approccio a directory decentralizzata è che non esiste più un server dedicato per il database della directory: il database è ora distribuito tra un insieme di pari (i capigruppo). Le dimensioni dei database sono relativamente piccole, dato che ogni capogruppo tiene traccia solo dei contenuti del suo gruppo. Inoltre, poiché l'informazione di directory in questo caso è distribuita tra normali pari, è più difficile per le azioni legali spegnere il sistema di condivisione di file P2P (a causa, per esempio, di violazione del copyright) chiudendo semplicemente un server di directory centralizzato.

Ci sono, comunque, un certo numero di svantaggi di questo approccio. Innanzi tutto, bisogna attivare un protocollo abbastanza complesso per costruire e mantenere aggiornata la rete sovrapposta. Per esempio, quando un capogruppo si disconnette, la rete sovrapposta deve rilevare la disconnessione e assegnare tutti i pari del capogruppo a nuovi capigruppo. Questa gestione della rete sovrapposta non è banale. Inoltre, sebbene questo approccio sia più decentralizzato di quello a directory centralizzata, i pari che sono capogruppo non sono esattamente equivalenti ai normali pari. In un sistema effettivamente decentralizzato e distribuito, tutti i pari sarebbero equivalenti! Poiché i capigruppo hanno più responsabilità, a essi è assegnata una porzione maggiore del lavoro, e quindi possono diventare colli di bottiglia. Infine, il sistema non è ancora del tutto privo di server, in quanto il nodo di bootstrap deve risiedere su un server sempre acceso. Sebbene le responsabilità dei nodi di bootstrap siano lievi (assegnare i pari ai capigruppo, designare i capigruppo, e mantenere aggiornata la rete sovrapposta), è comunque necessaria la presenza di uno o più nodi di bootstrap.

Inondazione di richieste (query flooding)

L'applicazione di condivisione di file P2P di dominio pubblico Gnutella [Gnutella 2002] usa un approccio completamente distribuito per fornire la localizzazione dei contenuti. In Gnutella, i pari prima si auto-organizzano in una rete sovrapposta. A differenza dalla rete sovrapposta per le applicazioni P2P a directory decentralizzata (come appena discusso), la rete sovrapposta di Gnutella ha una topologia piatta, non strutturata. Tutti i pari sono uguali; non esiste una struttura gerarchica con capigruppo. Una tale topologia è mostrata nella Figura 2.34. Un pari si aggrega alla rete come

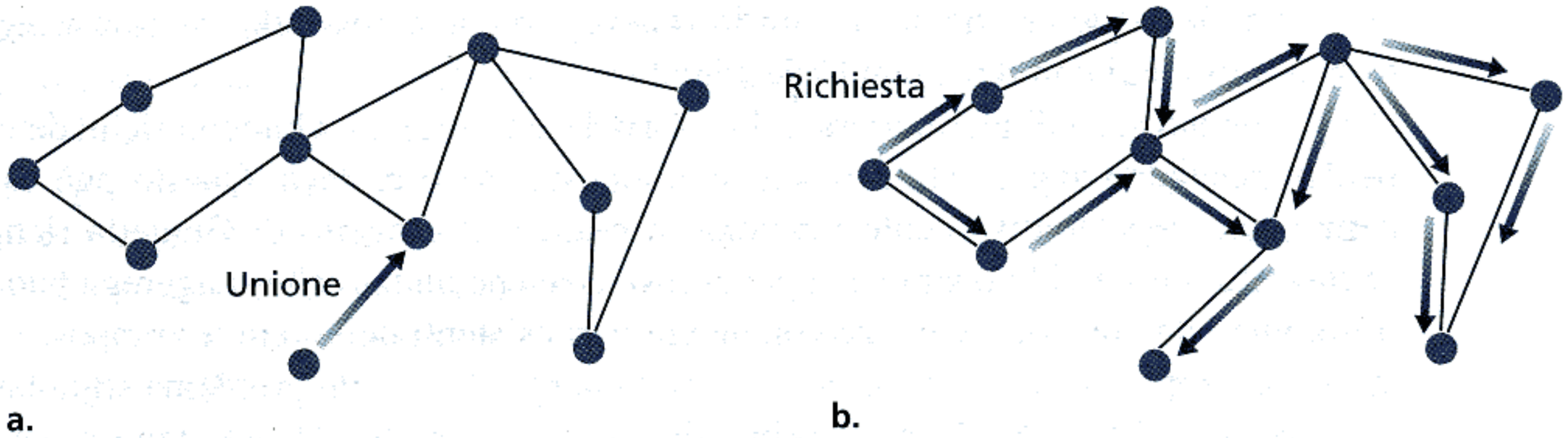

Figura 2.34 ♦ (a) il pari si unisce alla rete sovrapposta; (b) inondazione di richieste nella rete sovrapposta.

nel caso precedente, contattando un nodo di bootstrap, che comunica al pari l'indirizzo IP di uno o più pari facenti parte della rete sovrapposta. Ogni pari nella rete sovrapposta ha conoscenza solo dei pari suoi vicini, cioè, i pari con cui ha rami comunicanti nella rete sovrapposta. Come nel caso della rete sovrapposta gerarchica, è richiesto un protocollo abbastanza complesso per mantenere aggiornata la rete sovrapposta, dato che i pari si connettono e disconnettono spesso.

Per la localizzazione degli oggetti, Gnutella non usa directory (centralizzate o decentralizzate) ma piuttosto un'**inondazione di richieste** (*query flooding*). In particolare, quando un pari, diciamo Alice, vuole un oggetto, manda una richiesta dell'oggetto a ognuno dei pari suoi vicini nella rete sovrapposta. Inoltre, come mostrato nella Figura 2.34, ciascuno dei pari vicini di Alice rilancia il messaggio di richiesta a tutti i pari suoi vicini (esclusa Alice). La procedura continua, in modo che ogni pari nella rete sovrapposta riceve il messaggio di richiesta di Alice. Se un pari che riceve la richiesta ha una copia dell'oggetto desiderato, manda indietro una risposta al pari che ha originato la richiesta, indicando che possiede una copia.

L'architettura di Gnutella è elegante e attraente per diverse ragioni. Primo, tutti i pari hanno responsabilità simili; non ci sono capigruppo oberati di lavoro. Quindi l'architettura di Gnutella è fortemente decentralizzata. Secondo, nessuno dei pari memorizza informazioni di directory che mettono in relazione i contenuti con gli indirizzi IP. L'assenza di un database (centralizzato o distribuito) semplifica enormemente la progettazione.

D'altra parte, l'architettura di Gnutella è spesso criticata per non essere scalabile. In particolare, con il query flooding, ogni volta che un pari invia una richiesta, la richiesta si propaga a ogni altro pari nella rete sovrapposta. Quindi, viene immessa nella rete una grande quantità di traffico di richieste, e ogni pari viene bombardato da messaggi di richiesta.

I progettisti di Gnutella hanno risposto a questo problema introducendo un limite al raggio di propagazione delle richieste. In particolare, quando Alice invia il suo messaggio iniziale di richiesta, un campo contatore di nodo viene inizializzato a un limite specifico (diciamo 7). Ogni volta che il messaggio di richiesta raggiunge un nuovo pari, il pari decrementa il campo contatore di nodo prima di rilanciare la richiesta ai suoi vicini. Quando un pari riceve una richiesta con il campo contatore di nodo uguale a zero, non la inoltra. In questo modo, quando un nodo inizia una richiesta, l'inondazione è localizzata in una regione della rete sovrapposta. Questo approc-

cio riduce il traffico di richieste, con lo svantaggio che è possibile che non vengano trovati tutti i pari con il contenuto desiderato.

Come nel caso di architettura a directory decentralizzata, Gnutella richiede che i pari mantengano aggiornata una rete sovrapposta che li collega. Questo può essere complicato, ma effettivamente funziona in pratica. Il progetto di Gnutella richiede anche alcuni nodi di bootstrap sempre accesi, così che un pari che si aggrega può stabilire una relazione di vicinanza con alcuni pari esistenti nella rete sovrapposta. Costruire un'applicazione 2P2 senza alcun nodo di bootstrap è un problema stimolante!

Ma Gnutella, come Napster prima di essa, e KaZaA/FastTrack dopo di essa, è stata un'applicazione Internet di grande successo, che si è diffusa a letteralmente milioni di utenti nello spazio di pochi mesi. L'adozione sorprendentemente veloce ed estesa delle applicazioni di condivisione di file P2P, e il Web e l'instant messaging prima di esse, è una testimonianza della saggezza del progetto architetturale complessivo di Internet, un progetto che non avrebbe potuto prevedere il ricco e in continua espansione insieme di applicazioni Internet che sarebbero state sviluppate nei successivi 25 anni. I servizi di rete offerti alle applicazioni Internet – trasporto connectionless di datagram, trasporto affidabile connection-oriented di datagram, l'interfaccia socket, l'indirizzamento, e la traduzione di indirizzi (DNS), tra gli altri – si sono dimostrati in grado di permettere lo sviluppo di migliaia di applicazioni. Poiché queste applicazioni si poggiano tutte in cima agli esistenti quattro strati della pila protocollare di Internet, esse richiedono solo lo sviluppo di nuovo software client-server da utilizzare nei terminali. Questo, in sostanza, ha permesso a queste applicazioni di essere rapidamente sviluppate e adottate.

Ma non tutte le nuove applicazioni Internet hanno avuto un successo ampio come quello del Web, dell'instant messaging e della condivisione di file P2P. La conferenza multimediale e il trasferimento di flussi multimediali sono due applicazioni che devono ancora ottenere successo su larga scala. Questo può essere dovuto alla scarsità di servizi offerti dall'architettura di Internet odierna (esamineremo nel Capitolo 6 le estensioni all'architettura di Internet proposte per supportare queste applicazioni), o può essere dovuto a fattori socio-economici; si vedrà con il tempo.

2.10 Sommario

In questo capitolo abbiamo studiato gli aspetti sia concettuali sia di implementazione delle applicazioni di rete. Abbiamo visto l'ubiquità del paradigma client/server adottato dalle applicazioni Internet e il loro uso nei protocolli HTTP, FTP, SMTP, POP3 e DNS. Abbiamo studiato in alcuni dettagli questi importanti protocolli dello strato di applicazione, e le applicazioni a essi associate (Web, trasferimento di file, e-mail e il Domain Name System). Abbiamo esaminato come si può usare il socket API per costruire le applicazioni di rete, e abbiamo trattato non solo l'uso dei socket nel trasporto end-to-end con servizio orientato alla connessione (TCP) e senza connessione (UPD), ma abbiamo anche costruito un semplice server Web usando i socket. Infine, abbiamo concluso questo capitolo con una panoramica sulle tecniche attualmente in uso per la distribuzione di contenuti in una rete, esaminando il caching, le reti per la