

*A mio marito Marco
e alla mia famiglia*

SOMMARIO

Studi recedenti hanno mostrato che il consumo energetico legato a Internet rappresenta una porzione rilevante, e crescente, del consumo energetico complessivo della nostra società. Diventa quindi estremamente importante andare verso l'impiego di protocolli e applicazioni Internet efficienti dal punto di vista energetico. Il maggior contributo al consumo energetico è dovuto agli edge device di Internet (cioè PC e data center). Come esempio particolarmente significativo, in questa tesi consideriamo il fatto che gli utenti lasciano i propri PC continuamente accesi per soddisfare il requisito di connettività permanente richiesto dalle applicazioni Peer-to-Peer (P2P) per la condivisione di file, come BitTorrent, attualmente la più popolare piattaforma P2P di Internet. Per ridurre questo consumo energetico, senza però penalizzare la Quality of Service degli utenti BitTorrent, nella prima parte di questa tesi proponiamo una nuova architettura basata sull'introduzione di un proxy BitTorrent, Energy Efficient BitTorrent (EE-BitTorrent). Gli utenti BitTorrent delegano le operazioni di download al proxy e, in seguito, possono spegnere i propri PC mentre il proxy scarica i file ad esso richiesti. Abbiamo implementato la nostra soluzione e l'abbiamo validata in un testbed reale prendendo in esame uno scenario dipartimentale. I risultati sperimentali ottenuti mostrano che, rispetto all'approccio di BitTorrent convenzionale, la nostra soluzione è molto efficace nel ridurre il consumo energetico senza introdurre alcun degrado della QoS. In particolare, i nostri risultati mostrano che la soluzione proxy-based consente una riduzione del consumo energetico fino al 95 % e, allo stesso tempo, una riduzione significativa del tempo medio di download del file.

Dato inoltre che molti utenti BitTorrent accedono a Internet da reti residenziali, è importante ottimizzare le prestazioni degli utenti BitTorrent residenziali, anche in termini di consumo energetico. Quindi, abbiamo confrontato il protocollo BitTorrent legacy con EE-BitTorrent anche in uno scenario residenziale. Abbiamo mostrato che, in questo contesto, le prestazioni ottenute dall'utente sono fortemente influenzate dal throughput in uplink consentito dalla rete di accesso. In particolare, quando la velocità di uplink disponibile è bassa, il protocollo BitTorrent legacy ha scarse prestazioni ed EE-BitTorrent è migliore in termini di tempo medio di download e di consumo energetico sul PC dell'utente. Quando invece la velocità di uplink è buona, accade il contrario. Motivati da questi risultati, nella seconda parte di questa tesi abbiamo progettato e implementato AdaBT, un algoritmo adattivo che seleziona dinamicamente l'opzione BitTorrent più efficiente (tra quella legacy e quella proxy-based), in funzione delle condizioni operative sperimentate dall'utente. I nostri risultati sperimentali mostrano che AdaBT è in grado di ridurre significativamente il tempo di download ed il consumo energetico utilizzando BitTorrent legacy od EE-BitTorrent.

ABSTRACT

Recent studies have shown that the Internet-related energy consumption represents a significant, and increasing, part of the overall energy consumption of our society. Therefore, it is extremely important to look for energy-efficient Internet applications and protocols. The largest contribution to this energy consumption is due to Internet edge devices (PCs and data centers). As a particularly significant example, in this Ph.D. thesis we address the fact that users leave their PCs continuously powered on for satisfying connectivity requirements of Peer-to-Peer (P2P) file sharing applications, like BitTorrent, currently the most popular P2P Internet platform. To reduce these energy consumptions, without penalizing the Quality of Service of BitTorrent users, in the first part of this thesis we propose a novel architecture based on the introduction of a BitTorrent proxy, Energy Efficient BitTorrent (EE-BitTorrent). BitTorrent users delegate the download operations to the proxy and, then, power off their PC, while the proxy downloads the requested files. We implemented our solution and validated it in a realistic testbed considering a departmental scenario. Experimental results show that, with respect to the legacy BitTorrent approach, our solution is very effective in reducing the energy consumption without introducing any QoS degradation. Specifically, our results show that the proxy-based solution can provide up to 95% reduction in the energy consumption and, at the same time, a significant reduction in the average file download time.

Since most of BitTorrent users access the Internet from residential networks, it is important to optimize the performance of residential BitTorrent users, also in terms of energy consumptions. Thus, we compare the legacy BitTorrent protocol with EE-BitTorrent also in a residential scenario. We show that, in this environment, the performance achieved by the user is strongly influenced by the uplink throughput allowed on the access network. Specifically, when the available uplink rate is low, the legacy BitTorrent protocol performs poorly and EE-BitTorrent outperforms it in terms of average download time and energy consumption at the user's PC. The opposite occurs when the uplink rate is good. Motivated by these results, in the second part of this thesis we designed and implemented AdaBT, an adaptive algorithm that dynamically selects the most efficient BitTorrent option (i.e., legacy or proxy-based), depending on the operating conditions experienced by the user. Our experimental results show that AdaBT is able to reduce significantly the download time and the energy consumption choosing the legacy BitTorrent or EE-BitTorrent.

INDICE

CAPITOLO 1

INTRODUZIONE.....	1
-------------------	---

CAPITOLO 2

STATO DELL'ARTE SU ENERGY EFFICIENCY	4
2.1 TASSONOMIA SULL'ENERGY EFFICIENCY IN INTERNET	5
2.1.1 <i>Internet core</i>	5
2.1.2 <i>Data center</i>	15
2.1.3 <i>Edge device</i>	17
2.2 DISCUSSIONE	32

CAPITOLO 3

PROTOCOLLO BITTORRENT	37
3.1 DESCRIZIONE DEL PROTOCOLLO	37
3.2 FUNZIONAMENTO DEL PROTOCOLLO.....	42
3.3 DISCUSSIONE	43

CAPITOLO 4

EENERGY EFFICIENT BITTORRENT	45
4.1 MOTIVAZIONI	45
4.2 DEFINIZIONE DEL PROTOCOLLO EE-BITTORRENT	47
4.3 VALUTAZIONE SPERIMENTALE IN UNO SCENARIO DIPARTIMENTALE	49
4.3.1 <i>Analisi dell'efficienza energetica</i>	49
4.3.2 <i>Risultati sperimentali</i>	53
4.4 VALUTAZIONE SPERIMENTALE IN UNO SCENARIO RESIDENZIALE	59
4.4.1 <i>Analisi dell'efficienza energetica</i>	60
4.4.2 <i>Risultati sperimentali</i>	61
4.5 LEZIONE IMPARATA	67

CAPITOLO 5

ADAPTIVE BITTORRENT (ADABT)	69
5.1 ALGORITMO ADABT	70
5.2 VALUTAZIONE SPERIMENTALE DI ADABT	71

CAPITOLO 6

CONCLUSIONI	74
BIBLIOGRAFIA	76
APPENDICE A.....	81
RINGRAZIAMENTI.....	97

Capitolo 1

Introduzione

La diffusione su vasta scala di Internet e di ICT (Information and Communication Technology) dal 1990 ad oggi ha avuto importanti conseguenze, introducendo migliorie ed efficienza in molti processi manifatturieri. Per contro, il massiccio uso di Internet ha introdotto un nuovo significativo e crescente contributo al consumo energetico totale della nostra società. Molti studi mostrano che l'energia totale consumata quotidianamente dalle persone che utilizzano Internet per navigare, controllare le email, scaricare file e in generale, per svolgere le loro attività quotidiane, è già molto elevata e ci si aspetta aumenti ancora nel prossimo futuro, di pari passo con l'espandersi del ruolo di Internet nella società. Circa 74 TeraWatt per ora (TWh) all'anno di elettricità, che sono circa il 2 % - 3 % del consumo energetico totale degli Stati Uniti, vengono consumati da Internet. Inoltre, si stima che circa il 40 % di questa energia potrebbe essere risparmiata utilizzando tecniche di power management su dispositivi connessi a Internet [1]. La causa di questo spreco energetico è il comportamento degli utenti che sono tipicamente disinteressati e non prestano attenzione al problema energetico. Alcuni studi mostrano che, in ambienti quali grandi compagnie, uffici e dipartimenti, solo il 44 % dei computer, il 32 % dei monitor, e il 25 % delle stampanti vengono spenti durante la notte [2]. In ambienti residenziali, il consumo elettrico stimato di computer e monitor è stato circa 22 TWh all'anno nel 2007, che rappresenta il 2,7 % dell'intero consumo elettrico in ambito domestico nell'Unione Europea [3]. Considerando il core di Internet (rete di accesso e di trasporto), altri studi stimano un consumo energetico dell'1 %, che potrebbe aumentare fino al 4 % con l'aumentare della velocità di accesso [4]. Riguardo a server e data center, alcuni report affermano che il loro consumo è stato di circa 61 miliardi di kilowatt per ora (kWh) nel 2006, che è circa l'1,5 % del consumo totale di elettricità degli Stati Uniti e che corrisponde a 4,5 miliardi di dollari [5].

Le motivazioni che spingono a fare qualcosa per il problema energetico sono due: non solo ridurre il consumo energetico, quindi i costi, per operare con apparecchiature ICT, ma

anche ridurre le emissioni di CO₂ per vivere in un ambiente più sostenibile. Come mostrano alcuni report, l'intera industria ICT, che include PC, server, impianti di raffreddamento, telefonia, LAN, impianti di telecomunicazione per uffici e stampanti, è responsabile di circa il 2 % delle emissioni globali di CO₂, l'equivalente della porzione di cui è responsabile l'aviazione [6]. Specificamente, si parla di un totale di 300 milioni di tonnellate di emissioni di CO₂ per reti e dispositivi di rete, circa 400 milioni di tonnellate di emissioni di CO₂ per PC e periferiche, e circa 100 milioni di tonnellate di emissioni di CO₂ per i data center [7].

Sulla base di questi dati, la comunità dei ricercatori nel settore delle reti è interessata a ridurre il consumo energetico puntando sul green computing e sul green networking. Rendere efficiente l'uso di apparecchiature ICT potrebbe ridurre il consumo energetico causato da questi dispositivi e allungare il loro tempo di vita. L'obiettivo è un adeguato risparmio impiegando tecnologie green, cioè adattando le prestazioni e il consumo energetico all'utilizzazione e ai requisiti operativi invece che operare sempre a piena capacità indipendentemente dal carico di traffico. Questo approccio energy-aware potrebbe avere importanti conseguenze dal punto di vista ambientale ed economico. Recenti studi stimano che il risparmio energetico totale ottenibile utilizzando Green Network Technologies (GNT) è 1331 GWh per anno, che rappresenta circa il 68 %. In particolare, questo valore è da considerarsi suddiviso tra gateway, dove la riduzione è di circa 1060 GWh per anno, equivalente a circa il 70 %, e infrastrutture e dispositivi di telecomunicazione per i quali il totale guadagno sarebbe 271 GWh per anno, che corrisponde al 65 % [8].

L'interesse dei ricercatori tende a concentrarsi in particolare nell'area degli Internet Service Provider (ISP) e delle Internet company ed è quindi volto a ridurre i consumi relativi al core di Internet (ad esempio dei router) e ai data center. Minor attenzione è invece dedicata all'ottimizzazione del consumo di potenza nei Personal Computer (PC), anche se la quantità di energia consumata negli uffici e nelle case da questi dispositivi rappresenta la porzione maggiore del consumo energetico totale. Infatti, nel 2007 i data center in USA sono stati responsabili di un consumo di circa 2 TWh per anno, mentre i PC in ambienti domestici e aziendali sono stati responsabili di circa 16 TWh per anno [9]. Questo è principalmente dovuto all'elevato numero di PC utilizzati in tutto il mondo. Inoltre i PC sono tipicamente

gestiti da comuni utenti che spesso li lasciano costantemente accesi per dimenticanza o disinteresse. Lo spreco energetico dovuto a questo comportamento disinteressato potrebbe essere facilmente evitato spegnendo i PC tramite ad esempio un sistema centralizzato di spegnimento come NightWatchman [10]. Però tale soluzione potrebbe scontentare gli utenti. Infatti, molti PC vengono lasciati accesi intenzionalmente dagli utenti, sia in ambienti lavorativi che domestici, per eseguire attività di rete quali, ad esempio, la condivisione di file tramite Peer-to-Peer (P2P). Recenti studi [11] indicano che una grande porzione del traffico Internet attuale è P2P (40 % - 73 %). Tra le varie piattaforme P2P inoltre, BitTorrent è il protocollo più popolare, responsabile del 50 % - 75 % dell'intero traffico P2P. Quindi, concentrarsi su soluzioni per il green P2P rappresenta una sensibile direzione di ricerca verso un Internet maggiormente energy friendly.

Motivati da questi dati, in questa tesi proponiamo una versione di BitTorrent efficiente dal punto di vista energetico per uno scenario dipartimentale / aziendale ed una successiva estensione per uno scenario residenziale. Il resto della tesi è organizzato come segue: il capitolo 2 descrive una tassonomia che schematizza le varie tecniche di risparmio energetico in Internet illustrate in letteratura; il capitolo 3 illustra il protocollo BitTorrent tradizionale; il capitolo 4 presenta Energy Efficient BitTorrent, la soluzione proposta per ottenere efficienza energetica nel settore dei personal computer, in particolare per applicazioni P2P, e ne mostra l'implementazione e la valutazione sperimentale in ambito dipartimentale e in ambito residenziale, mentre la sua estensione per adattarla ad uno scenario residenziale (AdaBT) è riportata nel capitolo 5; infine, il capitolo 6 trae alcune conclusioni.

Stato dell'arte su energy efficiency

Per affrontare il problema energetico nelle reti e per evitare sprechi di energia, abbiamo bisogno di definire quali sono i punti in cui essa viene sprecata e in che modo, per quantificarla e valutare quanta energia potrebbe essere risparmiata. Dal punto di vista dei consumi energetici consideriamo il modello di rete rappresentato nella Figura 1, che mostra la ripartizione dei consumi di potenza per una rete generale.

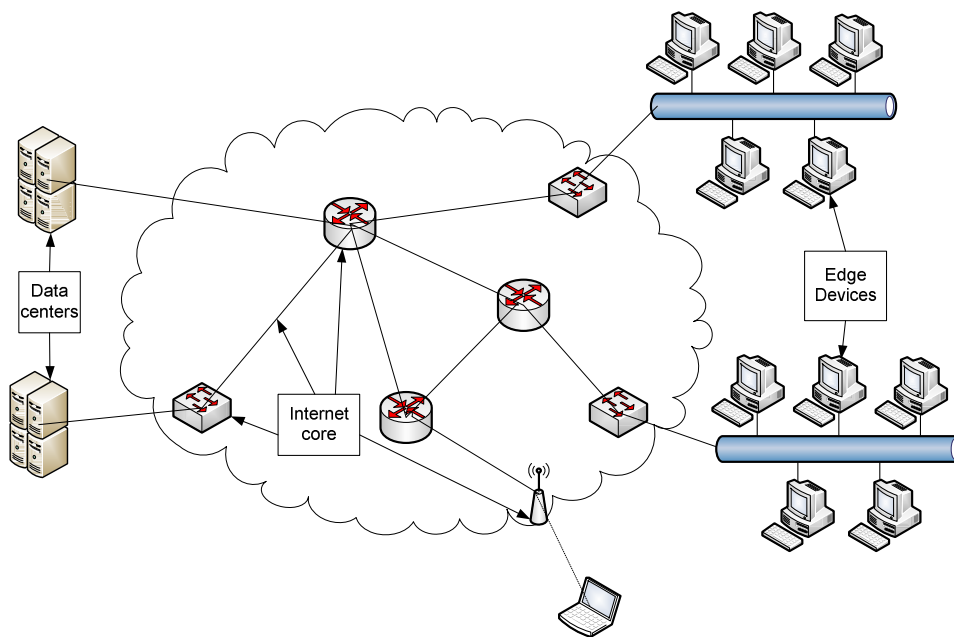


Figura 1 - Modello di rete.

Il nostro modello di rete è costituito da tre punti in cui l'energia viene consumata: elementi del *core di Internet* (router, switch, access point, link) che sono responsabili del consumo *diretto* di energia definito come l'energia utilizzata da collegamenti e apparecchiature di rete, ma non dai dispositivi di bordo; i *data center* (server), e gli *edge device*, che sono tipicamente PC, notebook, display, e che sono entrambi responsabili del consumo energetico *indotto* definito come la porzione di potenza consumata per lo stato di dispositivi che necessitano di mantenere la connettività di rete.

Sulla base del modello di cui sopra e della ripartizione di potenza, il consumo di energia, quindi le tecniche di conservazione dell'energia proposte in letteratura, potrebbero essere classificate in tre diverse classi: sprechi energetici a livello del core di Internet, dei data center e degli edge device. Vari approcci diversi possono essere sfruttati, anche contemporaneamente, per ridurre il consumo energetico delle reti. Ad un livello molto generale, possiamo identificare per ciascuna delle tre principali aree di spreco energetico nelle reti alcune tecniche di risparmio.

2.1 Tassonomia sull'energy efficiency in Internet

La Figura 2 mostra la tassonomia relativa alle tecniche di energy efficiency per Internet fornite dalla letteratura. In questo paragrafo esamineremo le principali tecniche per l'efficienza energetica in Internet, concentrandoci in particolare, dopo una breve carrellata sia sull'Internet core che sui data center, sugli edge device (ad esempio, PC e notebook) che essendo molto numerosi e diffusi rappresentano una fonte di consumo molto ampia. Tutti gli schemi che descriveremo appartengono ad una delle tre zone di sprechi energetici che abbiamo presentato.

2.1.1 Internet core

I componenti dell'Internet core sono collegamenti, router e switch e sono responsabili del consumo energetico *diretto* di Internet (secondo livello della tassonomia mostrata in Figura 2). In [4] gli autori stimano che il consumo di potenza dell'Internet core rappresenti l'1 % - 4 % del consumo elettrico nei paesi forniti di banda larga, variabile in funzione della velocità di accesso, e identificano come colli di bottiglia dal punto di vista energetico i router ed i collegamenti non in fibra ottica.

Come mostrato dal terzo livello della tassonomia in Figura 2, il consumo energetico diretto dovuto agli elementi dell'Internet core può essere causato da tre diversi elementi: (i) collegamenti, (ii) dispositivi di rete (router e switch), e (iii) protocolli e hardware di rete non energy-aware. Di conseguenza, le tecniche di efficienza energetica possono agire (i) a livello fisico (*Link Level*), (ii) a livello di trasporto (*Device Level*) o (iii) a livello di routing (*Network Level*).

A livello fisico (i), per i collegamenti, la velocità aumenta costantemente, e con essa aumenta il consumo energetico relativo. Le tecniche di risparmio energetico messe in atto sono tre, come evidenziato al quarto livello della tassonomia in Figura 2: (a) *speed scaling*, (b) *link aggregation*, e (c) *new physical encoding*.

Tecniche di (a) *speed scaling* si basano sul principio di proporzionalità e adattano la velocità del collegamento alle esigenze di traffico attuali. Misure di potenza hanno dimostrato che l'energia consumata dai collegamenti dipende fortemente dalla velocità di trasmissione dati supportata. Misure sperimentali effettuate su link Ethernet [12] hanno dimostrato che non vi è quasi alcuna differenza, in termini di consumo di energia, nel passaggio dal 10 Mbps al 100 Mbps. Invece, il consumo energetico del collegamento aumenta di circa 4 W quando si passa da 100 Mbps a 1 Gbps, e molto di più (circa 10 W - 20 W) quando si passa da 1 Gbps a 10 Gbps. Queste cifre sono molto importanti data l'attuale tendenza verso tassi di collegamento sempre più elevati. È anche importante sottolineare che il consumo energetico di un link non dipende dalla sua utilizzazione, cioè, link completamente inattivi e link utilizzati a piena capacità consumano circa la stessa potenza. Dall'altro lato, i collegamenti hanno tipicamente una bassa utilizzazione [13] e sono caratterizzati da lunghi periodi di inattività durante i quali non vengono trasmessi dati. Per esempio, nei collegamenti Ethernet per la connessione a Internet dei dispositivi di una rete aziendale o universitaria l'utilizzazione media del collegamento è nell'ordine del 1 % - 5 % [14], il che significa che i link rimangono inattivi per la maggior parte del tempo. Considerazioni analoghe valgono anche per i collegamenti ADSL che sono ampiamente usati in Europa per l'accesso residenziale a Internet tramite banda larga.

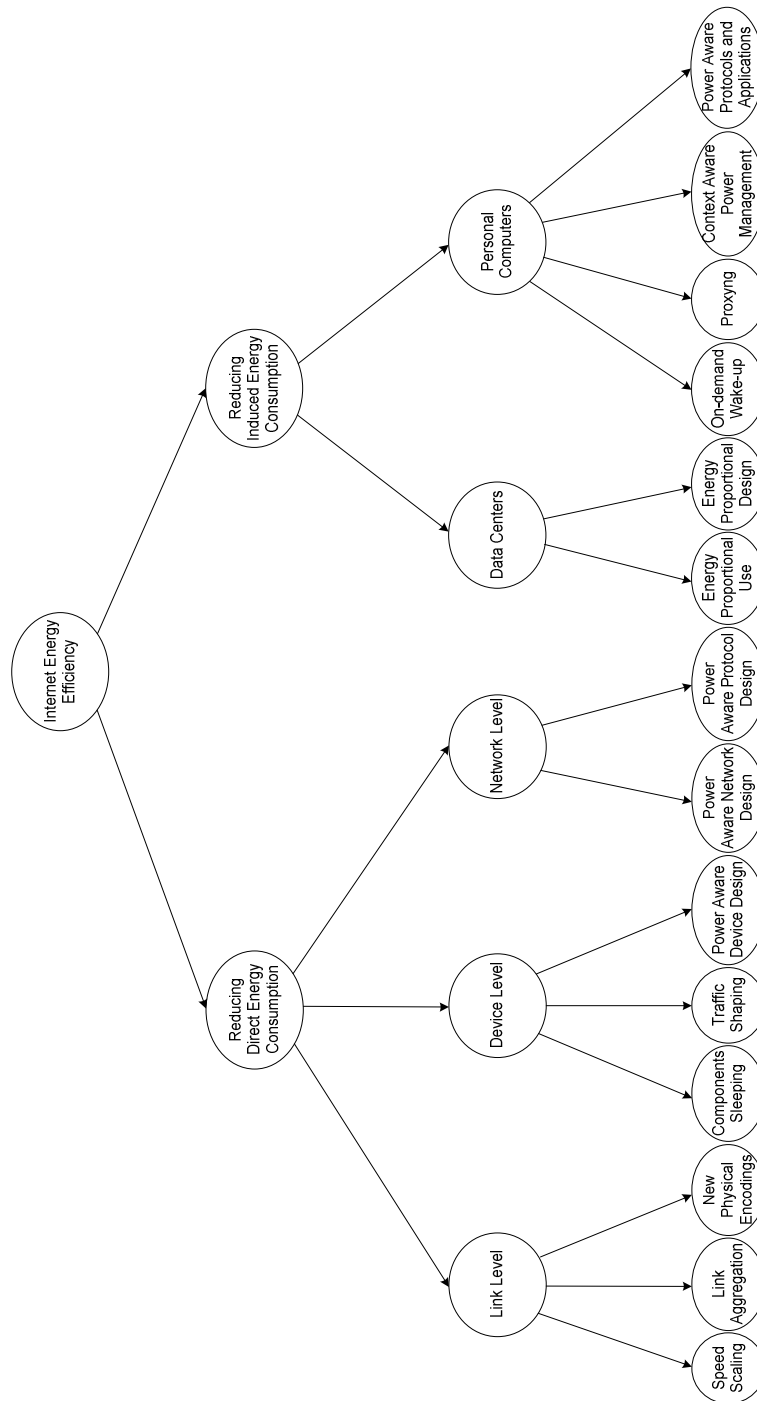


Figura 2 - Tassonomia generale.

Pertanto, un significativo risparmio energetico potrebbe essere ottenuto rendendo il loro consumo energetico proporzionale alla loro utilizzazione, ad esempio, adattando la velocità dei dati che viaggiano sul link alle esigenze del traffico [9].

[9, 15, 16] propongono l'idea di una velocità di collegamento adattiva, *Adaptive Link Rate* (ALR), per adeguare la velocità e quindi il consumo all'utilizzazione del link. ALR è attualmente lo standard della 802.3az IEEE Energy Efficient Ethernet (EEE) task force [17]. ALR è destinato principalmente a ridurre il consumo energetico dei link Ethernet di confine (ad esempio, tra un PC e uno switch). Essa considera i tassi di velocità attuali (10 Mbps, 100 Mbps, 1 Gbps e 10 Gbps), e consente agli end-point del collegamento di negoziare automaticamente e concordare la velocità minima compatibile con l'utilizzazione corrente del collegamento. La sua applicazione in dispositivi reali richiede la presenza di schede di rete (NIC) che siano in grado di commutare tra diversi livelli di power management con diversi stati di consumo di potenza e due ulteriori componenti, cioè, un meccanismo per cambiare la velocità dei dati da un valore ad un altro, e una politica per decidere quando cambiare la velocità [18]. Un meccanismo di commutazione rapida della velocità chiamato Rapid PHY Selection (RPS) [19] è basato su un two-way handshake seguito da una fase di re-sincronizzazione tra gli end-point. Un'evoluzione di tale meccanismo di commutazione attivo / inattivo è LPI (Low-Power Idle) [20]. L'algoritmo utilizzato invece per decidere quando variare la velocità è il Dynamic Ethernet Link Shutdown (DELS) [21, 22].

Un'altra tecnica per il risparmio energetico a livello di link è il meccanismo di (b) link aggregation, che consiste nell'adattare dinamicamente il numero di link necessari in base alla variazione del volume di traffico (Dynamic Link Control). In [23] e [24] gli autori propongono un sistema per il controllo della capacità di trasmissione tra gli switch utilizzando Link Aggregation (LA), un meccanismo standardizzato in IEEE 802.3ad e utilizzato per aumentare la velocità di trasmissione tra gli switch. Il Link Aggregation Control Protocol (LACP) è utilizzato tra gli switch per la creazione di un unico collegamento logico aggregando più collegamenti fisici. Gli autori di [23] e [24] hanno modificato il LACP con l'aggiunta di alcune caratteristiche, in particolare le misure e la stima del traffico in arrivo. Nel dettaglio, il LACP misura il traffico in entrata, valuta il numero di collegamenti fisici necessari per soddisfare la domanda di traffico, comunica tale

numero allo switch posto dall'altra parte del collegamento e viene stabilito il nuovo collegamento logico. La stima della capacità di trasmissione si basa sul valore del throughput di picco.

L'ultima tecnica di risparmio energetico considerata, a livello di link, è il (c) new physical encoding. [25] propone un nuovo meccanismo di codifica ad alta efficienza energetica a livello di collegamento. In particolare, suggerisce un modello per le codifiche Ethernet e una codifica migliore per 100 Mbps Ethernet. Per la codifica Ethernet, due componenti svolgono il ruolo principale per il consumo di energia, cioè i circuiti di codifica e il sistema di trasmissione. Nel modello energetico proposto l'energia spesa dal circuito di codifica è molto più grande di quella necessaria per la trasmissione dell'energia in una tipica Ethernet. L'idea di base è quindi quella di minimizzare il budget di energia necessaria per la codifica a livello fisico, data la larghezza di banda, il tasso di errore e la velocità di trasmissione dei dati.

A livello di trasporto (ii), la letteratura suggerisce tre tecniche di risparmio energetico per router, switch e access point, come evidenziato al quarto livello della tassonomia in Figura 2: (a) *component sleeping*, (b) *traffic shaping*, e (c) *power-aware design*.

Le tre tecniche sono spesso usate contemporaneamente e sono basate su principi comuni. Dal momento che il consumo di energia è dovuto al tempo in cui i dispositivi restano attivi e di non utilizzo, e poiché il tempo di utilizzo effettivo di un dispositivo è molto basso rispetto al tempo che rimane acceso, l'idea è mettere le interfacce di rete e i componenti di router e switch a riposo. Come riportato in [26], nel 2000 negli Stati Uniti il consumo causato da elementi dell'Internet core (router, switch, access point e hub) è stato 6 TWh, corrispondente allo 0,07 % della spesa energetica totale negli Stati Uniti.

L'approccio generale proposto in [26] suggerisce che tecniche di (a) *component sleeping*, che consentono cioè di porre in stato sleeping interfacce di rete e componenti di router / switch, siano una strategia praticabile per il risparmio energetico, ma che richiedono modifiche alle specifiche del protocollo corrente. [26] definisce due differenti approcci: lo sleeping non coordinato, un approccio a livello di link, in cui un'interfaccia è posta in stato dormiente sulla base di decisioni solo locali, e lo sleeping coordinato, un approccio a livello di rete in cui il protocollo di routing aggrega il traffico in pochi percorsi (in situazioni di basso carico) per poter porre in modo esplicito alcune interfacce in stato di riposo. [26]

suggerisce anche risposta al problema di come l'architettura ed i protocolli utilizzati in Internet possono essere modificati per raggiungere l'obiettivo di efficienza energetica. Per gli switch, IEEE 802.1d necessita di cambiamenti per permettere alle interfacce di essere messe in stato sleeping senza attivare la ricostruzione di spanning tree e la disabilitazione dello scambio periodico di pacchetti spanning tree. Per i router, sono necessarie modifiche in OSPF per evitare di trattare i collegamenti dormienti come link che falliscono, per limitare i messaggi 'Hello' soltanto per il risveglio delle interfacce dormienti e per introdurre un meccanismo di previsione per identificare i momenti di basso carico e sapere quando poter andare in stato sleeping. È inoltre necessario un algoritmo per identificare il numero minimo di collegamenti necessari per soddisfare le esigenze di QoS dei flussi supportati. [22] e [27] propongono lo sleeping delle schede di rete durante i periodi di bassa utilizzazione del link o in periodi di inattività e prevedono quanti pacchetti arriveranno in un periodo di tempo sufficiente a spegnere il collegamento. In particolare, utilizzano lo sleep timer programmabile di cui molte schede di rete Intel sono fornite e che di solito è alimentato a batteria, per cambiare periodicamente lo stato di alimentazione quando il collegamento è inattivo o sotto-utilizzato. Essi assumono un semplice modello di collegamento, in cui entrambe le interfacce operano in due modalità, cioè spenta o alla velocità massima, quindi, se il collegamento è attivo o spento da un lato allora sarà nello stesso stato anche all'altra estremità. Si suppone la presenza di un buffer di trasmissione in cui vengono memorizzati i pacchetti quando il collegamento è spento. Lo stato di sleeping non è possibile se il tempo tra due pacchetti è inferiore al tempo di ri-sincronizzazione e al tempo di transizione tra lo stato off e on del link. Gli autori hanno inoltre sviluppato un algoritmo per la previsione dell'arrivo dei pacchetti e per decidere quando e per quanto tempo spegnere il collegamento, conosciuto come protocollo On / Off [22] o Dynamic Ethernet Link Shutdown (DELS) [27]. L'algoritmo gira sull'interfaccia del trasmettitore che informa l'interfaccia del ricevitore dell'intervallo di tempo per cui sarà in stato sleeping, che dipende dall'occupazione del buffer e dal tempo di inter-arrivo tra i pacchetti. Con questo meccanismo, [22] pone in modalità low-power le interfacce Ethernet a entrambe le estremità del collegamento per lungo tempo (40 % - 98 %) e suggerisce di mettere in stato sleeping contemporaneamente altre interfacce dello stesso switch (37 %). Inoltre, il ritardo sperimentato è nel range di 0,1 - 1,5 ms, il che è abbastanza piccolo da non influenzare il

comportamento dei protocolli di livello superiore. Gli stessi autori hanno proposto un modello di sleeping per le schede di rete degli switch per consentire lo spegnimento durante i periodi di basso traffico per tipi di interfaccia diversi, cioè interfacce di collegamento tra switch e host ed interfacce di collegamento tra switch e altri switch [26]. La scelta di mettere in stato sleeping singole porte invece che l'intero switch è dovuta al tempo di interattività, derivato da studi che mostrano che il traffico è suddiviso in blocchi di circa 1 s nel primo tipo di collegamento, con la possibilità di sleeping, e inferiore a 1 s nel secondo tipo di collegamento. Il modello di sleeping proposto si basa su diversi livelli di funzionalità e si compone di cinque stati: Off, Simple Sleep, HAS (Hardware Assisted State), HABS (Hardware Assisted Buffered State) e Wake. Ognuno di questi stati è associato ad un diverso consumo energetico e ad un diverso rendimento in termini di prestazioni: l'utilizzo o meno di un buffer per memorizzare i pacchetti che, se il buffer non viene utilizzato vanno persi, l'utilizzo di pacchetti che possano svegliare l'host oppure il semplice utilizzo di uno sleep timer, ecc. Lo stato Simple Sleep, che non utilizza il buffer, consuma meno energia rispetto allo stato HAB, ma introduce il 7,5 % di perdita di pacchetti.

Le tecniche di (b) traffic shaping permettono di modellare il traffico consentendo la presenza di periodi di idle nei dispositivi in cui possono, quindi, essere posti in stato sleeping. PAUSE Power Cycle (PPC) è un metodo basato sull'uso di PAUSE, l'algoritmo MAC per il traffic shaping, per la creazione di periodi di inattività in cui mettere a riposo l'intero switch e non solo le schede di rete [28]. Il meccanismo MAC PAUSE di 802.3 supportato da tutte le schede di rete Ethernet compatibili con lo standard IEEE 802.3, consiste nella possibilità di inviare frame per bloccare l'host a monte dall'inviare pacchetti per un certo periodo di tempo (PAUSE quanta). Si tratta di un meccanismo di controllo di flusso che provoca un ciclo di stati on-off sui collegamenti Ethernet. PPC, con questo meccanismo, permette ad uno switch LAN di mettere in pausa periodicamente tutti i collegamenti attivi e spegnere per tutto il tempo i link che sono in pausa. La quantità di energia risparmiata con questo meccanismo dipende da quanto tempo è trascorso in stato Off e dal compromesso tra risparmio energetico e perdita di pacchetti che può verificarsi durante tali momenti. Il meccanismo PPC ha un effetto trascurabile su protocolli e applicazioni. Una seconda versione di PPC è un meccanismo PPC adattivo, che può adattare il duty-cycle on-off all'utilizzazione del collegamento. Utilizza la storia del carico

di traffico per stimare il carico futuro osservando quanti byte ha ricevuto nell'ultimo periodo On per determinare se deve mettere in pausa o meno. Se il numero di byte è maggiore di una soglia stabilita, il periodo di Off viene saltato.

Le tecniche di (c) power-aware design prevedono la progettazione dei dispositivi di rete di nuova generazione secondo principi di efficienza energetica. Dato che l'equivalente ACPI non esiste per i router hardware, [29] propone di introdurre funzionalità avanzate di gestione dell'alimentazione e tecnologie green nell'architettura dei dispositivi di rete di accesso e di trasporto per le reti di nuova generazione (Next Generation Networks). A livello network, il consumo energetico è suddiviso tra le reti di trasporto e il core (36 %) e la componente di accesso (64 %), che rappresenta la componente maggiore. La nuova generazione di Software Router (SR) basata su sistema operativo Linux e hardware COTS (Commercial Off-The-Shelf), è in grado di sfruttare sistemi multi-CPU e multi-core, e include le più avanzate funzionalità di gestione dell'alimentazione. La capacità di elaborazione dei dispositivi di rete è modulata in risposta all'utilizzazione della rete, adattando le prestazioni al carico di traffico o sfruttando i periodi di inattività per andare in stato sleeping. Questi meccanismi sono realizzati a livello hardware spegnendo componenti o scalando la frequenza e la tensione di funzionamento e, a livello software, da un'applicazione di controllo (governor), che dinamicamente configura il profilo di potenza da utilizzare. L'obiettivo è ridurre al minimo il consumo di energia durante i periodi d'inattività e raggiungere un compromesso tra prestazioni e consumi. L'idea di traffic shaping è utilizzata anche in questo campo, nei SR, e forza la CPU tra stato sleeping e poi attiva a pieno regime, in modo da ottenere risparmio di energia a prezzo della latenza. L'obiettivo è raggiunto raggruppando i pacchetti che viaggiano a velocità lenta e mandandoli a velocità di picco raccolti in burst attraverso il SR.

Il consumo energetico dei router aumenta con l'aumentare della popolarità di Internet e della banda disponibile. Mentre, grazie al progresso tecnologico, la potenza per ogni byte trasmesso diminuisce della metà ogni due anni, la velocità delle linee aumenta in modo esponenziale e aumenta quindi la densità di potenza. I tradizionali sistemi di raffreddamento ad aria per i router stanno diventando inadeguati e i sistemi di raffreddamento a liquido, che possono essere necessari, sono più costosi. Fatte queste considerazioni, la consapevolezza relativa al consumo energetico come principio guida

nella progettazione e configurazione delle reti e nella progettazione e realizzazione dei protocolli di routing rappresenta un altro significativo approccio orientato al risparmio di potenza nell'ambito del core di Internet (Network Level). In particolare, a livello di routing (iii), la letteratura suggerisce due tecniche di risparmio energetico, come evidenziato al quarto livello della tassonomia in Figura 2: (a) *power-aware network design*, e (b) *power-aware protocol design*.

La prima categoria (a) rappresenta l'insieme di tecniche di progettazione power-aware per ridurre il consumo energetico della rete, sia impiegate dai produttori di router sia dai progettisti di rete. L'obiettivo della progettazione power aware è ridurre al minimo il numero di router in rete, rispettando però i requisiti di robustezza e prestazionali. In particolare vengono utilizzati due approcci: i sistemi multi-chassis e i sistemi alternativi basati su tecnologia ottica. Il primo approccio permette di raggruppare insieme diverse schede di rete per formare un singolo router logico che le collega ad uno switch con chassis non-bloccante scalabile. In questo modo, anche se il consumo di potenza globale aumenta, il carico di calore si suddivide rendendo efficace anche l'utilizzo di sistemi di raffreddamento ad aria. Il secondo è uno switch ottico, che fornirebbe maggiore larghezza di banda con una dissipazione di potenza molto bassa. La limitazione di tale soluzione è però l'evoluzione non completa della tecnologia ottica [30].

[30] propone due approcci: topologie di rete multi-router, che soddisfano gli obiettivi progettuali di capacità e robustezza con router a basso consumo che sostituiscono quelli ad alta potenza, e la progettazione di rete orientata a limitare le operazioni costose in termini di potenza a un sottoinsieme di router. Questi principi sono stati sviluppati commutando tra accensione e spegnimento di nodi e link sotto vincoli di connettività e QoS con un algoritmo euristico in [31] e con un approccio centralizzato in [32]. L'idea di [31] è spegnere i nodi e i link sottostando a vincoli di QoS (utilizzo massima del collegamento) essendo nota la topologia della rete e la richiesta di traffico. Gli autori hanno implementato due algoritmi diversi: uno orientato al nodo e uno orientato al link. Poi hanno combinato i due approcci euristici in una soluzione unica, che prima tenta di disattivare i nodi e, in una seconda fase, cerca di spegnere i collegamenti. Questo approccio si basa sul fatto che la disattivazione di un nodo è più difficile, ma il risparmio energetico che ne deriva è molto più grande rispetto al caso in cui si disattivi il link. Con questi semplici

algoritmi euristici si riesce a disattivare il 10 % dei nodi e il 25 % dei link, pur garantendo un'utilizzazione che rimane al di sotto dell' 80%. [32] propone l'Energy Management System (EMS), un approccio centralizzato che gestisce dinamicamente le reti cablate per ridurre al minimo il consumo di energia mantenendo la QoS. EMS è un meccanismo software che funziona a livello IP e che commuta dinamicamente tra on e off i link e i nodi in risposta al carico di traffico garantendo i requisiti di QoS attraverso il cambiamento dei percorsi seguiti dal traffico. Il sistema monitora costantemente i flussi di traffico, lo stato del nodo e il consumo di energia, e seleziona la configurazione di rete che offre il miglior livello di QoS col minor consumo energetico. Esso gestisce inoltre i cambiamenti dinamici nei link e nei nodi e reindirizza il traffico.

Grandi reti hanno percorsi paralleli per raggiungere una destinazione per garantire robustezza e bilanciamento del carico. Principi energy-aware possono essere applicati anche a livello di protocollo di routing (b), con tecniche di progettazione che tengano conto dei consumi di potenza e consentano di disattivare i percorsi inutilizzati durante i periodi di basso carico [30]. Un altro approccio, proposto in [33], è un meccanismo di inoltro sincrono o pipeline per i pacchetti IP. La condizione necessaria è avere lo stesso riferimento di clock in tutti i router, che è realizzato da un timer globale o UTC (Coordinated Universal Time) e che viene utilizzato per l'inoltro ottimale del traffico periodico. [33] propone due sottoreti parallele, indipendenti e integrate: Internet e una super-rete ad alta velocità per l'inoltro dei pacchetti IP in pipeline. Gli switch sono sincronizzati utilizzando un time frame (TF), un contenitore virtuale di pacchetto IP. Il tempo è diviso in cicli in cui i TF sono raggruppati e inoltrati in blocco. Ne risulta una trasmissione di pacchetti che si muovono in rete come in una pipeline. I vantaggi di questo approccio sono molteplici: l'inoltro in pipeline offre la complessità ottimale per la progettazione degli switch e introduce semplificazione nella progettazione dei router, cioè senza buffer, senza necessità di elaborazione delle intestazioni dei pacchetti, meno riconfigurazione della struttura di commutazione (ogni ciclo e non ogni pacchetto), un minor numero di elementi di switch, utilizzazione del link superiore pur mantenendo basso ritardo, e l'impiego di tecnologia ottica, con un potenziale molto alto per il risparmio energetico. È possibile migrare questa rete virtuale nella Internet esistente come una rete parallela, per trasportare video e altri contenuti ad alta richiesta di banda.

[34] esplora infine il concetto di Energy Aware Routing (EAR) che utilizza il profilo energetico dei router, cioè l'Energy Profile Aware Routing (EPAR), con le informazioni di consumo energetico di tali apparecchi, quando instradano pacchetti. Sono definiti una serie di profili energetici (PE), cioè il consumo di energia dato in termini di carico di traffico di un componente di rete (lineare, Log_{10} , cubico, On-Off, e Log_{100}). L'EAR è un meccanismo efficiente di riduzione dei consumi energetici che prende decisioni di routing in funzione di informazioni di carico reali e fornite dal EPAR.

2.1.2 Data center

I data center sono responsabili, come gli edge device, di parte del consumo energetico *indotto* di Internet (secondo livello della tassonomia mostrata in Figura 2), ed in generale rappresentano l'1,5 % del consumo energetico totale negli Stati Uniti.

Il tempo di risposta dei Web server è un parametro molto importante in questo tipo di sistemi e determina la qualità del servizio. Per evitare ritardi nella risposta ai client, non si utilizza un singolo server che risponde alle richieste dei client, ma cluster di server che contengono un elevato numero di macchine proporzionale al picco di lavoro e sempre attive. In effetti, quindi, per la maggior parte del tempo queste sono sotto-utilizzate. Quindi, i cluster di server lavorano per la maggior parte del tempo a pieno regime, ovvero con il consumo energetico massimo, ma in condizioni di carico leggero. I server presentano un profilo di utilizzazione diverso da altri dispositivi di rete quali laptop e PC: mentre gli ultimi sperimentano lunghi periodi di inattività, è raro che i server siano completamente inattivi e operano per la maggior parte del tempo al 10 % - 50 % dei livelli massimi di utilizzazione [35].

Infatti, i server devono essere disponibili, anche durante i periodi di basso carico, in quanto le informazioni e la capacità computazionale sono distribuite tra essi, e perché spesso eseguono molte piccole operazioni in background, rendendo impossibile l'uso di tecniche idonee per il risparmio energetico impiegate per altri dispositivi. Per poter utilizzare tecniche di risparmio energetico è necessario identificare prima le condizioni di traffico a cui i server sono sottoposti.

Come mostrato al terzo livello della tassonomia in Figura 2, la soluzione generale per ottenere efficienza energetica nei data center sembra essere in concetto di *energy*

proportional che prevede che il consumo energetico sia proporzionale alla quantità di lavoro svolto dal server stesso, e che idealmente non dovrebbe consumare energia quando è inattivo. Questo principio può essere applicato a due livelli: (i) nell'uso di server già esistenti, e (ii) nella progettazione delle nuove macchine destinate a tale scopo.

Nel primo caso (i) si adotta il meccanismo *energy proportional use*, ossia utilizzare il numero minimo di server in grado di rispondere alle richieste con un certo livello di qualità del servizio. Questa soluzione può essere implementata in modo centralizzato o in modo distribuito. In particolare, [36] propone il bilanciamento della potenza, cioè la proporzionalità tra il consumo energetico ed il carico. Gli autori propongono di mantenere attivi solo un numero minimo di server in un cluster, necessari per servire tutte le richieste in momenti di basso carico, mentre gli altri server possono essere messi in stato low-power. Quando il carico di lavoro aumenta, i server dormienti vengono svegliati tramite un meccanismo di wake-up (che tratteremo in seguito). Questo approccio è basato su un meccanismo di sleeping e wake-up per ottenere risparmio energetico in situazioni di basso carico, ma consente di usufruire della completa funzionalità in caso di carichi elevati. L'approccio proposto rappresenta un compromesso tra latenza e risparmio energetico. Mentre i server sono ora orientati a minimizzare la latenza e consumano una grande quantità di energia utilizzando tutti i server in stato di piena capacità, [36] propone una soluzione orientata a minimizzare il consumo di energia e pagando questo vantaggio in termini di latenza che è più elevata a basso carico perché i server attivi, che sono solo un sottoinsieme dei server nel cluster, devono gestire tutte le richieste.

Nel secondo caso (ii), cioè la progettazione di macchine efficienti dal punto di vista energetico, si considera invece un altro aspetto messo in evidenza in [35]. Il picco dell'efficienza energetica dei server occorre al picco della sua utilizzazione e decresce linearmente mentre l'utilizzo diminuisce. Considerando la gamma di utilizzo tipica dei server (10 % - 50 %), queste macchine non lavorano quindi mai in condizioni ottimali di efficienza energetica. Questo accade perché attualmente i server sono progettati in modo orientato all'ottimizzazione della gestione dei picchi di carico. Per questa ragione, [35] propone un nuovo paradigma di progettazione per lo sviluppo dei server, l'*energy proportional design*.

L'obiettivo è una progettazione dei componenti hardware delle macchine server orientata al carico medio invece che al carico di punta, sulla base della proporzionalità tra consumi energetici e la quantità di lavoro eseguito per ottenere risparmio energetico. Tutti i componenti devono operare in modalità low-power quando sono attivi, consumando solo una piccola percentuale della loro potenza di picco. La gamma di potenza dinamica riportata in [35] per la CPU è il 70% della potenza di picco, per le DRAM è inferiore al 50 %, per le unità disco è del 25 % e per gli switch di rete è del 15%. A titolo di esempio, un server con una gamma di potenza dinamica del 90% potrebbe consumare la metà dell'energia, seguendo una nuova curva di efficienza energetica: alta efficienza energetica (80 %) per il 30 % di utilizzo, che è il valore tipico, e un consumo di energia che aumenta linearmente con l'utilizzo. Utilizzando componenti hardware progettati seguendo i suddetti principi, i server potrebbero evitare l'utilizzo di software di power management per ottenere il risparmio energetico.

Attualmente, le CPU sono gli unici componenti per cui esistono profili che mostrano principi di energy-proportional, mentre per gli altri componenti, ad esempio memoria e disco, non ce ne sono. [35] suggerisce una caratterizzazione completa delle misure di sistema in condizioni non-peak e i principi di proporzionalità energetica come obiettivo nella progettazione di componenti e nello sviluppo di sistemi.

2.1.3 Edge device

Gli edge device sono responsabili, come i data center, di parte del consumo energetico *indotto* di Internet (secondo livello della tassonomia mostrata in Figura 2). Essendo molto diffusi e numerosi, abbiamo focalizzato la nostra analisi su questi dispositivi, ed in particolare sul consumo energetico dovuto ad essi e sulle tecniche utilizzate per ottenere risparmio. Per chiarezza riportiamo in Figura 3 un estratto della tassonomia che riguarda gli edge device.

Anche se il consumo energetico dei dispositivi personali è limitato (circa 100 W per PC e circa 30 W per notebook) in confronto ai router e ai data center, tuttavia essi sono molti diffusi e numerosi quindi l'energia totale consumata dai dispositivi personali in ambienti domestici e in ufficio è molto grande. I PC danno di conseguenza anche un importante contributo alla produzione di CO₂. Inoltre i dispositivi personali sono tipicamente utilizzati

con scarsa o nessuna attenzione per il problema energetico. Per esempio, molti PC nelle case e negli uffici vengono lasciati accesi, anche di notte e nel week-end, solo per mantenere la connettività di rete. I costi complessivi per la connettività di rete sono difficili da stimare. Soltanto per gli Stati Uniti il consumo è compreso tra 2 e 5 miliardi di dollari l'anno [37]. Infatti, come riportato da Bruce Nordman in [38], nel corso del 2007 negli Stati Uniti i data center hanno rappresentato circa 2 TWh all'anno di consumo energetico, mentre i dispositivi di case e uffici corrispondevano ad un consumo di circa 16 TWh all'anno. Inoltre, gli utenti in genere non applicano alcuna politica di risparmio energetico.

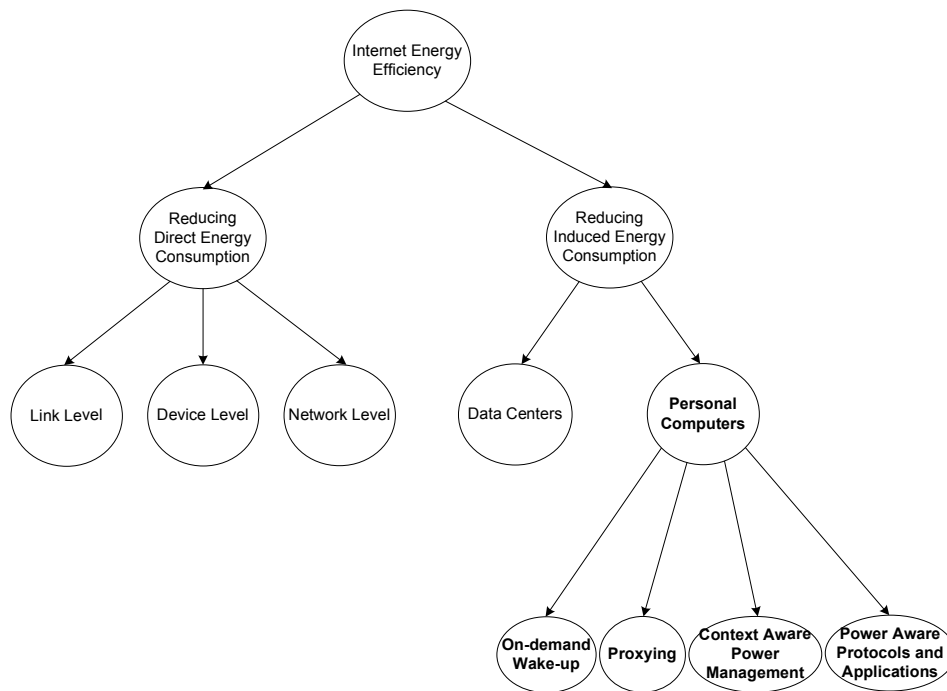


Figura 3 - Tassonomia parziale relativa agli edge device.

Questo emerge con chiarezza, ad esempio, nel PC Energy Report dalla UK National Energy Foundation [10] relativo al consumo energetico dei PC utilizzati dagli utenti sul lavoro. In particolare, questo rapporto evidenzia che circa il 21 % dei computer utilizzati sul posto di lavoro non vengono quasi mai spenti (durante la notte e nei weekend), e questo provoca sprechi di energia di circa 1,5 TWh di elettricità all'anno (corrispondenti a circa 700.000 tonnellate di CO₂). Lo spreco di energia dovuto agli edge device è in parte causato dagli

utenti che lasciano il PC sul posto di lavoro sempre acceso per pigrizia o dimenticanza. In questo caso il PC dovrebbe essere semplicemente spento e la letteratura suggerisce soluzioni per lo spegnimento centralizzato, come NightWatchman [10]. Al contrario, il caso più interessante è il caso in cui un PC rimane sempre acceso per l'esecuzione di alcune attività di rete come, ad esempio, l'esecuzione di applicazioni P2P per la condivisione di file. Dal momento che gli edge device sono utilizzati solo per una piccola frazione del tempo che sono accesi [39], la maggior parte dell'energia consumata per mantenere la connettività potrebbe essere risparmiata con l'introduzione di adeguati meccanismi per la gestione dell'alimentazione. Per essere efficaci, questi meccanismi devono risparmiare energia senza introdurre degrado significativo in termini di prestazioni. Le tradizionali tecniche di power management sono incompatibili con tale aspetto, ma studi relativi al traffico di rete, ad esempio [40, 14], permettono di identificare i periodi idle durante i quali è possibile risparmiare energia spegnendo i PC o mettendoli in stato sleeping. Nello specifico, abbiamo bisogno di ideare meccanismi che possono consentire ad un PC di stare in stato sleeping durante i tempi di inattività, al fine di risparmiare energia, e di riprendere tempestivamente le funzionalità operative ogni volta che un pacchetto esterno viene ricevuto, in modo da ridurre al minimo l'impatto sulla capacità di risposta del sistema. Come mostrato in Figura 3, per gli edge device, che identifichiamo come Personal Computer possono essere posti in stato sleeping e risvegliati quando la rete ha bisogno di un'interazione diretta utilizzando (i) tecniche di *on-demand wake-up* per mezzo di un messaggio speciale [41]. Per sopperire all'interazione coi protocolli per i semplici eventi di routine o per altri scopi relativi alla connettività è possibile sfruttare tecniche di (ii) *proxying*, che permettono di delegare ad un componente o ad un'altra macchina (proxy) le interazioni minime con la rete del PC dormiente e di consentirgli da remoto di risvegliare i PC dormienti tramite meccanismi di wake-up spracitati. Inoltre, l'enorme diffusione di display contribuisce ulteriormente al consumo energetico. I display possono essere spenti quando nessuno si trova alla postazione PC, quindi tecniche di (iii) *context aware power management* possono essere utilizzate per questo scopo. Infine, è possibile ottenere un ulteriore contributo al risparmio energetico progettando il software in modo da ottenere (iv) *protocolli e applicazioni power-aware*. Analizziamo brevemente le tecniche di on-demand wake-up, context aware power management e di progettazione di protocolli e applicazioni

power-aware per riservare una trattazione più ampia alle tecniche proxying, dal momento che la nostra risposta al problema del consumo energetico fa parte di questa categoria di soluzioni.

2.1.3.1 On-demand wake-up

Per quanto riguarda le tecniche di (ii) on-demand wake-up, Wake-on-LAN (WoL) è un meccanismo di Ethernet che consente al PC dormiente di essere svegliato da remoto con un messaggio speciale chiamato Magic Packet [41]. Il Magic Packet è tipicamente inviato da un altro computer sulla stessa LAN del PC di destinazione, ma potrebbe anche essere inviato da qualsiasi computer in Internet. Per funzionare, il WoL prevede che una parte della scheda di rete Ethernet debba restare sempre attiva. Pertanto, un PC inattivo può essere messo in una delle tante modalità a basso consumo disponibili per risparmiare energia (per esempio, sleeping, ibernate, Soft-off), ma non può essere scollegato dalla fonte di alimentazione o deve utilizzare una fonte alternativa di alimentazione (cioè un batteria). Questo introduce un consumo energetico di stand-by che è, però, molto più piccolo del consumo di energia del PC in modalità attiva. Una scheda di rete speciale è necessaria sul PC dormiente, cioè dotata di una'alimentazione ausiliaria esterna per consentire un funzionamento parziale, di un collegamento con la linea di interrupt per il risveglio e di funzionalità di rilevazione e di ricezione dei Magic Packet. Quando il meccanismo di WoL è attivo, la scheda di rete Ethernet intercetta i Magic Packet, cioè speciali frame MAC contenenti sedici ripetizioni dell'indirizzo MAC del PC bersaglio. Il Magic Packet viene inviato all'indirizzo di broadcast della sottorete del PC di destinazione ed è, quindi, ricevuto da tutte le schede NIC collegate a quella sottorete. Tuttavia, è scartato da tutti i PC tranne quello il cui indirizzo MAC corrisponde all'indirizzo specificato. Dopo aver ricevuto un Magic Packet valido la scheda di rete riattiva il PC. Poiché il meccanismo di WoL si basa su indirizzi MAC, in linea di principio il Magic Packet dovrebbe essere originato da un computer appartenente alla stessa LAN del PC di destinazione. Tuttavia, questa limitazione può essere facilmente rimossa permettendo ad un PC di essere svegliato da un qualsiasi computer su Internet. Ciò può essere ottenuto, ad esempio, utilizzando una rete privata virtuale (VPN) in modo che il computer remoto appaia come un membro della stessa LAN

del PC dormiente. In alternativa, il Magic Packet potrebbe essere un messaggio UDP destinato a una porta specifica e instradato attraverso Internet come un normale datagram. Sono state proposte diverse estensioni per il meccanismo di base WoL sopra descritto. Per esempio, le schede di rete Intel permettono diverse opzioni: Wake on Direct Packet, Wake on Magic Packet, Wake on Magic Packet from Power-off state, e Wake on Link [42]. Purtroppo, nella maggior parte dei casi, l'attivazione di questi meccanismi fa sì che si verifichino risvegli frequenti del PC rendendo così inutile il meccanismo di sleeping. Infatti, possono verificarsi risvegli inutili, che spreca energia. Inoltre, dato che il PC consuma molta più energia durante la transizione da riposo a risveglio piuttosto che durante le operazioni in stato stazionario, un elevato numero di riattivazioni non necessarie potrebbe portare addirittura a ulteriori sprechi. Riattivazioni inutili potrebbero essere evitate filtrando le richieste di riattivazione con altri meccanismi (vedi NCP di cui tratteremo in seguito).

2.1.3.2 Proxying

Un proxy (ii) è un'entità che può rispondere alle interazioni di rete minima per i dispositivi dormienti. In particolare, il dispositivo stabilisce che è il momento di entrare in modalità sleeping, comunica la sua decisione al proxy e gli passa il proprio stato. Il proxy mantiene la presenza di rete dell'host dormiente, determina quando quest'ultimo deve essere risvegliato e, una volta che l'host è sveglio, gli ripassa lo stato.

Perdere la connettività di rete sembra essere la più grande barriera all'utilizzo di stati di sospensione nei PC e nei dispositivi di bordo, e il proxy potrebbe far risparmiare più della metà dell'energia utilizzata. Per i PC desktop molto tempo viene speso in stato attivo e idle, ed il potenziale risparmio per un singolo PC desktop utilizzando tecniche di sleeping è di decine di dollari l'anno, mentre per 50 milioni di questi corrisponderebbe ad una cifra tra 0,8 e 2,7 miliardi di dollari all'anno [38].

L'idea di utilizzare un proxy per il risparmio energetico non è nuova. Infatti, architetture proxy-based sono state impiegate per garantire un accesso ad Internet efficiente dal punto di vista energetico per dispositivi mobili. Comunque, in quel caso, l'architettura proxy era progettata per supportare un dispositivo mobile che eseguiva un'applicazione client-server standard [43]. Più recentemente, l'idea di un'architettura proxy-based è stata proposta per

implementare soluzioni energy-aware in Internet, ad esempio in [38]. L'idea è utilizzare un proxy che prenda il posto dell'host nel rispondere alle minime interazioni della rete e che svegli l'host solo quando la rete richiede la sua interazione attiva. Quando ciò accade, è necessario un meccanismo di wake-up per svegliare l'host, per esempio utilizzando la Wake On LAN NIC citata nel sottoparagrafo precedente. Il proxy può essere sulla NIC del PC dormiente oppure può essere un dispositivo esterno. Tipicamente le soluzioni proxying sono usate insieme a soluzioni di on-demand wake-up.

[44] formalizza il concetto generale di proxy introducendo la *selective connectivity*. Questo meccanismo permette agli end system di scegliere il loro stato di connettività e di andare in stato sleeping per il risparmio energetico, ma continuando a mantenere la presenza in rete. In particolare, gli autori definiscono tre stati del sistema: acceso, spento e dormiente. Quest'ultimo non ha caratteristiche statiche, ma sotto il controllo dell'end system potrebbe variare basandosi sulle politiche e sui desideri dell'utente o sull'ambiente. Le attività sono classificate anche in base alle risorse necessarie, in un range compreso tra attività a bassa potenza e attività ad alta potenza, che corrispondono rispettivamente ai livelli di connettività da disconnesso a collegato a piena potenza. [44] suggerisce alcune componenti architetturali per supportare la selective connectivity. Gli *assistant* sono entità che potrebbero aiutare l'host mentre è in stato sleeping, eseguendo operazioni solitamente gestite da esso. L'assistant analizza il traffico in entrata diretto verso l'host per risvegliarlo basandosi su una politica stabilita dallo stesso. L'assistant potrebbe anche inviare messaggi di errore lieve, come "riprovare più tardi" verso l'host remoto in modo che questi possa capire che si verifica una temporanea e non permanente indisponibilità a comunicare. Potrebbe essere utile rendere noto lo stato di selective connectivity tra diversi layer e per gli host interconnessi in modo da comunicare che la ragione della momentanea non-disponibilità è l'efficienza energetica. In questo modo, per favorire il risparmio energetico, l'altra parte potrebbe decidere se cambiare l'interazione con l'host. A questo proposito, gli autori propongono due tipi nuovi di stato: stato Proxyable, in cui l'assistant fa le veci dell'host, e lo stato Limbo, in cui l'host scambia informazioni con gli host remoti per permettere loro di sapere che è in stato sleeping e non disconnesso. Il controllo basato sull'host permette all'utente di stabilire alcune politiche per gli host remoti che comunicano

con lui mentre è in modalità di selective connectivity. In particolare l'host decide come gli host remoti dovrebbero trattarlo e quali attività dovrebbe essere delegate ad altri.

Le soluzioni proxying si possono ulteriormente suddividere in (a) soluzioni proxy per uno specifico protocollo o applicazione, di cui anche la nostra soluzione fa parte, e (b) soluzioni proxy general purpose.

2.1.3.2.1 Application Proxy

Il proxying è una tecnica molto comune nel distributed computing. Tradizionalmente, (server) proxy sono stati utilizzati in applicazioni distribuite per migliorare le prestazioni di sistema (ad esempio il tempo di risposta) e ridurre il traffico di rete. Più recentemente, i proxy sono stati considerati nel campo del mobile computing per far fronte ad un certo numero di fattori tra cui le limitate capacità computazionali e le scarse risorse energetiche del dispositivo mobile, la mobilità dell'utente e la connettività intermittente o scarsa. In particolare, un (client) proxy è usato come surrogato del client mobile nella rete fissa, permettendo poi al dispositivo mobile di essere temporaneamente disconnesso dal sistema e risparmiare energia [45].

L'architettura UPnP Low Power [46] rappresenta un esempio di soluzione proxy per uno specifico protocollo. Le specifiche UPnP (Universal Plug and Play), rilasciate dal UPnP Forum, permettono ai dispositivi di connettersi automaticamente e formare spontaneamente reti, ad esempio per scambio di dati, intrattenimento, installazioni software, ecc. L'architettura UPnP legacy [47] si basa su un protocollo di discovery distribuito che richiede che tutti i dispositivi restino sempre accesi per rispondere ai messaggi di discovery. L'architettura UPnP low power definisce un Low Power Proxy per consentire ai dispositivi della rete UPnP di stare in stato sleeping e continuare a rispondere ai messaggi di discovery del punto di controllo UPnP.

Un'altra applicazione del meccanismo di proxying unito ad un meccanismo di on-demand wake-up è il SIP Catcher [48], un sistema che permette ai telefoni IP di restare in stato sleeping per lungo tempo, mentre il PC a cui sono connessi è anch'esso in stato sleeping. A causa della loro grande diffusione (20 milioni di utenti in USA), i telefoni IP, che consumano circa 10 W - 20 W, sono responsabili di spreco di energia dato che essi rimangono costantemente attivi ma vengono utilizzati per pochissimo tempo. SIP è il

Session Initiation Protocol ed è utilizzato per connettere telefoni IP in Internet. Specificamente, un utente (*caller*) si registra al SIP server e quando vuole chiamare il telefono IP invia un messaggio *invite* per contattarlo. Il SIP server localizza il telefono IP e trasmette il messaggio di *invite* al telefono IP destinatario. Questo risponde con il messaggio *trying* e poi con un messaggio *ringing*, dopodichè suona. Poi, il caller ed il telefono IP chiamato possono comunicare. Se il telefono IP chiamato è in stato *sleeping*, risulta non raggiungibile. Il SIP catcher è un sistema che gira sull'ultimo router in prossimità del telefono IP e che fa da proxy per le chiamate SIP. In particolare, quando esso rileva un messaggio *invite* da un caller per il telefono IP dormiente, lo sveglia e, quando quest'ultimo si è riattivato, gli inoltra il messaggio, e risponde al posto del telefono IP al protocollo SIP inviando un messaggio *trying*. Poi, il telefono IP invia al caller un messaggio *trying* e un messaggio *ringing*, completando l'handshake del protocollo SIP.

Anche le applicazioni P2P (come BitTorrent o Gnutella) sono state ideate supponendo che i PC restino costantemente attivi. Recenti studi [11] indicano che il 40 % - 73% del traffico Internet è P2P, quindi queste applicazioni sono responsabili di gran parte del consumo energetico di Internet e possono essere rese energy friendly grazie all'utilizzo di tecniche di proxying accoppiate a tecniche di wake-up remoto. [49] propone appunto un meccanismo di power management basato su proxy per Gnutella, che sfrutta anche il meccanismo WoL. Gnutella utilizza un meccanismo di flooding per trovare file nella overlay network. Definisce cinque tipi di messaggi che sono Ping, Pong, Query, Query Hit e Push. I messaggi Query e Query Hit sono utilizzati rispettivamente per trovare file e per rispondere alle richieste. Il power management proxy è un microcontrollore a basso consumo con capacità limitata che dovrebbe consumare molta meno energia rispetto all'host e che è posto sulla Ethernet NIC dell'host P2P o nello switch della sua stessa LAN. Il proxy rileva richieste di file dirette all'host dormiente e lo sveglia. Poi, l'host può fornire il file. Per prendere il posto dell'host in stato *sleeping*, il proxy deve condividere con esso informazioni di stato (*sleeping* o attivo), la lista di IP dei vicini dell'host dormiente e la lista dei nomi dei file condivisi dall'host. Rispetto all'host dormiente, il P2P proxy supporta un sottoinsieme di funzionalità: può iniziare e accettare connessioni coi vicini, ricevere ed inviare messaggi Query, inviare messaggi Query Hit e svegliare l'host in stato *sleeping*, ma non può né fornire né scaricare file. Quando l'host va in stato *sleeping* le sue connessioni

coi vicini vengono chiuse da esso e ristabilite dal proxy. Quando il proxy deve svegliare l'host accade il contrario. Il proxy utilizza il meccanismo WoL per svegliare l'host.

Anche la nostra soluzione si inserisce in questa categoria, cioè è una soluzione proxy-based per il risparmio energetico per la piattaforma P2P BitTorrent, che non utilizza però alcun meccanismo di risveglio.

Per completezza citiamo inoltre le soluzioni proxy-based in [50, 51] per l'efficienza energetica nella condivisione di file che utilizzano una versione semplificata di BitTorrent da dispositivi mobili. L'obiettivo principale in questo settore è il risparmio energetico volto a aumentare il più possibile la durata delle batterie del dispositivo mobile. In [51] viene proposto e valutato l'impiego di un proxy eseguito su un router a banda larga a casa dell'utente, mentre in [50] la proposta precedente è estesa considerando un proxy distribuito su un certo numero di router residenziali.

2.1.3.2 Network Connectivity Proxy

Gli application proxy possono essere utilizzati in ogni applicazione di rete e permettono di risparmiare molta energia, proporzionale al numero di utenti che la utilizzano, in particolare se girano su macchine che devono già rimanere costantemente accese per altre ragioni (ad esempio macchine che offrono servizi multipli). In ogni caso, essi sono specifici dell'applicazione, perciò è necessario un diverso proxy per ogni applicazione. Inoltre, essi richiedono tipicamente l'intervento umano, non sono cioè trasparenti all'utente. Idealmente, un PC dovrebbe entrare in stato sleeping in modo trasparente ogni volta che è inattivo, per risparmiare energia. Allo stesso tempo dovrebbe però apparire connesso e pienamente operativo per gli altri dispositivi di rete. Questo massimizzerebbe il risparmio energetico mentre minimizzerebbe, allo stesso tempo, l'impatto sulle prestazioni delle applicazioni di rete. Questo obiettivo può essere ottenuto utilizzando un *Network Connectivity Proxy* (NCP), cioè un'entità capace di mantenere la presenza in rete al posto del PC dormiente, gestendo tutti i pacchetti ad esso destinati intanto che si trova in stato sleeping.

Il concetto di NCP fu originariamente proposto in [52] per le reti Ethernet e poi esteso in articoli successivi [12, 15, 38] per reti IP in generale. Per mantenere la connettività durante i periodi di inattività, un PC esegue una serie di azioni, ad esempio risponde a richieste ARP periodiche, genera richieste DHCP periodiche per mantenere il suo indirizzo IP,

risponde a messaggi ICMP (ad esempio, richieste di ping), accetta connessioni TCP rispondendo a segmenti TCP SYN, e, più in generale, gestisce in modo appropriato tutti i pacchetti in arrivo. Un'analisi dettagliata dei pacchetti ricevuti da un PC durante i periodi di inattività, e i relativi protocolli, è riportata in [12] e, più recentemente in [53]. Il secondo considera sia ambienti domestici che lavorativi. Noti il tipo e la percentuale di pacchetti ricevuti, questi si possono classificare in funzione delle classi di azioni che essi richiedono e si possono così definire i requisiti del NCP [37, 53]. Di base, una volta ricevuto un pacchetto destinato al PC dormiente, il NCP dovrebbe eseguire una delle seguenti azioni: (i) rispondere direttamente al pacchetto; (ii) scartare il pacchetto; re-direzionare il pacchetto ad un altro PC (attivo) per processarlo; (iv) svegliare l'host e passargli il pacchetto per farlo processare adeguatamente; o (v) mettere il pacchetto in coda per ritardarne l'elaborazione a quando l'host si riattiverà. Inoltre, il NCP potrebbe essere istruito a generare richieste periodiche in veste del PC dormiente, ad esempio richieste DHCP per mantenere l'indirizzo IP [37]. In accordo col modello NCP descritto, possono essere investigate molte varianti [53]. In realtà, lo spazio di progettazione è abbastanza esteso dato che sono possibili molte soluzioni diverse in vari aspetti tra cui la complessità (insieme di funzionalità implementata dal proxy), il grado di trasparenza (possibili differenze nel comportamento di utente o applicazione con e senza il proxy), il posizionamento (luogo in cui si trova fisicamente il proxy, ad esempio, sullo stesso PC, su router / firewall, su un PC separato, ecc.), e l'implementazione (ad esempio, dispositivo collegato alla scheda di rete / scheda madre del PC, periferica esterna collegata tramite USB).

In [53] gli autori considerano quattro proxy diversi con diversa complessità, e confrontano le loro prestazioni in termini di risparmio energetico. Propongono anche un'architettura NCP generale e flessibile che può ospitare diverse scelte progettuali e presentano un semplice prototipo di essa. In questa implementazione il NCP si presume essere posizionato su una macchina stand-alone che è, quindi, responsabile del mantenimento della connettività di rete di più PC appartenenti alla stessa LAN. Nell'analisi delle performance vengono considerate le seguenti quattro varianti di Proxy: *Proxy-1*, che scarta tutti i pacchetti classificati come ignorabili, e sveglia il PC per la gestione di tutti gli altri pacchetti; *Proxy-2*, che scarta tutti i pacchetti classificati come ignorabili, risponde direttamente a pacchetti di protocolli che richiedono un'interazione minima, e sveglia il PC

per tutti gli altri pacchetti; *Proxy-3*, che esegue le stesse azioni del *proxy-2*, ma è più selettivo in quanto sveglia il PC dormiente solo quando il pacchetto ricevuto appartiene a un set di applicazioni specificato dall'utente; *Proxy-4*, che esegue le stesse azioni del *proxy-3* rispetto ai pacchetti in entrata, inoltre può essere incaricato di svegliare il PC per eseguire un'attività pianificata come il backup di rete, aggiornamenti antivirus, aggiornamenti software, e così via. Il confronto delle prestazioni si basa su tracce derivate da misure reali su pacchetti ricevuti da un insieme di PC durante i periodi di inattività, sia in una ambiente domestico che aziendale. I risultati mostrano che *Proxy-1* è inadeguato in ambienti d'ufficio e marginalmente adeguato negli ambienti domestici; *Proxy-3* fornisce un significativo risparmio energetico in ambiente domestico e negli scenari aziendali (il PC può restare in stato sleeping per la maggior parte del tempo di inattività), mentre le prestazioni di *Proxy-2* dipendono in gran parte dall'ambiente specifico. Infine, le prestazioni di *Proxy-4* sono vicine a quella del *Proxy-3* dato che le operazioni programmate sono generalmente poco frequenti.

Somniloqui [54] è un NCP di classe PDA proposto dai ricercatori Microsoft. Esso consiste in un dispositivo esterno, connesso al PC tramite porta USB e che include un processore low-power capace di eseguire un sistema operativo embedded, una memoria flash per la memorizzazione dei dati mentre il PC si trova in stato sleeping, e una o più interfacce di rete per comunicare con l'esterno. Quando il PC è in stato sleeping, la connettività è mantenuta dalla NIC del proxy (dato che quella del PC è disattivata). Il proxy realizza un meccanismo di filtraggio dei pacchetti, che applica filtri sotto forma di espressioni regolari, per ricevere pacchetti e svegliare il PC in caso di matching. Questo consente di gestire applicazioni di rete come shell sicura da remoto, richieste di accesso ai file, e chiamate VoIP, anche quando il PC è in stato sleeping. Inoltre, *Somniloqui* può agire anche come application proxy per alcune comuni applicazioni di rete come l'instant messaging ed il P2P file sharing. Ciò si ottiene mediante l'implementazione di una versione leggera della specifica applicazione (*stub*) sul proxy. L'applicazione *stub* consente al proxy di gestire autonomamente la maggior parte delle azioni richieste dall'applicazione, e di svegliare il PC solo quando si verificano eventi complessi. Un'implementazione prototipo di *Somniloqui* basata sulla piattaforma Gumstix è stata realizzata in [55]. Il prototipo è equipaggiato con un processore XScale a 200 MHz con 16 MB di memoria flash e 64 MB di RAM, un cavo

Ethernet (o wireless Wi-Fi), scheda di rete per la connettività, e due porte USB (una per lo sleeping / wake-up del PC, l'altra per l'inoltro dei dati ricevuti dalla scheda di rete al PC), ed esegue una versione embedded di Linux che supporta lo stack TCP / IP completo. Per dare un'idea della quantità di energia risparmiata da Somniloqui si può considerare che un comune PC consuma circa 100 W in funzionamento normale, e meno di 5 W quando viene utilizzato con Somniloqui. Pertanto, il risparmio energetico è nell'ordine del 95 % [54].

Somniloqui e le diverse varianti proxy considerate in [53] non sono in grado di preservare le connessioni TCP tra i momenti di attività e di sleeping del PC. Invece, questo aspetto è specificamente affrontato in [37]. Oltre ai compiti di cui sopra (cioè, cancellare i pacchetti ignorati, rispondere direttamente ai pacchetti che richiedono azioni minime, e svegliare l'host quando necessario), il NCP proposto in [37] è anche in grado di mantenere le connessioni TCP e i flussi di dati UDP. Questo risultato è ottenuto splittando la connessione TCP sul proxy, che si assume sia eseguito sulla stessa rete del PC (ad esempio, su un router) e può, quindi, coprire diversi PC. I pacchetti TCP destinati ad un dato PC sono bufferizzati a livello locale presso il NCP quando il PC è in stato sleeping, e trasmessi più tardi, quando il PC è di nuovo attivo. L'accodamento dei pacchetti per l'elaborazione successiva può effettivamente avere senso per alcune applicazioni di rete come l'instant messaging e l'accesso remoto sicuro (SSH). In [37] gli autori presentano un prototipo di loro NCP in esecuzione su un router Linksys WRT 54G.

[56] propone anch'esso l'idea di uno *sleep proxy* e Apple ha rivendicato il suo diritto al brevetto dello standard ECMA, anche se l'idea è molto simile all'idea NCP proposta nel 1998 in [52] e sopra descritta. In particolare essi suggeriscono un sistema per l'implementazione dello sleep proxy che funziona come spiegato di seguito. Come primo passo, il sistema riceve la registrazione di tutti i dispositivi in rete che offrono alcuni servizi. Ogni volta che un dispositivo va in stato di sleeping, comunica al sistema che si sta entrando in modalità risparmio energetico e il sistema mantiene un elenco di dispositivi per i quali lo sleep proxy dovrebbe agire. Quando il sistema riceve una richiesta per un servizio fornito da un dispositivo nell'elenco, verifica se lo sleep proxy può rispondere. In tal caso, il proxy risponde alla richiesta a nome del dispositivo dormiente. Altrimenti, il sistema invia al dispositivo un pacchetto di wake-up che determina l'uscita dallo stato di risparmio energetico e il dispositivo può direttamente rispondere alla richiesta.

[57] propone *SleepServer*, un sistema di rete proxy-based che permette agli host di andare in stato sleeping, ma restando raggiungibili a livello applicazione. Esso non richiede alcuna modifica dell'infrastruttura di rete o hardware addizionale, ma soltanto degli agenti software installati sugli host. Infatti, è un meccanismo completamente implementato via software, utilizzando tecniche di virtualizzazione. In particolare, l'architettura proposta è fisicamente costituita da una o più macchine SleepServer (SSR) sulla stessa sottorete degli host deleganti (ma possono comunque fungere da proxy per PC su sottoreti diverse grazie al meccanismo delle VLAN). Ogni SSR funge da proxy per un insieme di host e contiene le loro immagini sotto forma di Virtual Machine (VM), che mantengono la presenza degli host in rete quando questi sono in stato sleeping. Sul SSR è installato un *SSR-Controller*, che è un componente software che gestisce la creazione delle immagini degli host, la comunicazione tra gli host e le loro immagini, l'allocazione delle risorse, e la condivisione di queste tra gli host garantendo isolamento tra le immagini. Ogni host ha anche un componente software, il *SSR-Client*, che si connette allo SleepServer indicando il proprio indirizzo MAC e indirizzo IP e la sua configurazione di firewall. Prima di entrare in stato sleeping, esso invia lo stato delle sue applicazioni e tutte le sue porte TCP e UDP aperte al SSR-Controller che crea l'immagine dell'host sulla macchina SSR. L'immagine utilizza la stessa configurazione di rete e i parametri dell'host per cui è proxy. In questo modo, mentre l'host è in stato sleeping, l'immagine interagisce con la rete al posto di esso. Se l'interazione dell'host dormiente fosse necessaria, il SSR-Controller sveglia l'host e disabilita la sua immagine sullo SleepServer. SleepServer introduce un significativo risparmio energetico di circa il 60 % - 80 % ed è particolarmente consigliato per ambienti con LAN aziendali con un elevato numero di host. Supporta inoltre sistemi operativi eterogenei.

Infine, lo Standard ECMA-393 [58] è lo standard, adottato nel Febbraio 2010, che definisce il proxy per il mantenimento della connettività di rete e la presenza di host che si trovano il stato sleeping per prolungati periodi di tempo allo scopo di risparmiare energia. Lo standard definisce il comportamento e l'architettura del proxy. In particolare, specifica la capacità del proxy, lo scambio di informazioni tra il proxy e gli host, il comportamento del proxy per 802.3 Ethernet e 802.11 WiFi, ed anche i comportamenti operativi incluso rispondere ai pacchetti, generare pacchetti, ignorare pacchetti e svegliare gli host.

2.1.3.3 Context-Aware Power Management

Le tecniche di (iii) context-aware power management sono relative al consumo in particolare delle periferiche, che rappresentano un elemento di spreco aggiuntivo nel contesto degli edge device. In particolare, i display sono la maggior fonte di consumo nei laptop. Le tecniche di Context Aware Power Management (CAPM) sono un insieme di metodi per rilevare l'interazione dell'utente con il PC tramite sensori a basso consumo e, in caso di assenza dell'utente, per l'impiego di tecniche di power management di sistema.

[59] propone un metodo CAPM che utilizza una videocamera per rilevare la presenza dell'utente davanti al monitor e che spegne il monitor se l'utente non è presente. Il sistema è equipaggiato con un *image detector* che cattura le immagini dalla telecamera. L'immagine viene processata da un *face detector* che rileva la presenza dell'utente. Quando quest'ultimo non è presente il sistema spegne il display.

[60] è un lavoro sui metodi CAPM nelle WLAN, che rileva i click del mouse su una GUI e li raggruppa in cluster per determinare il contesto. Ad ogni cluster è associato un livello di attività di rete relativo, da alto a basso e sulla base di esso il sistema attiva o disattiva i sistemi di comunicazione.

Un altro approccio è quello di rilevare la posizione relativa dell'utente rispetto al PC per determinare se lo sta utilizzando oppure no. [61] propone questo meccanismo come metodo CAPM, che si basa sulla posizione dell'utente come informazione di contesto e che spegne il PC quando questo non viene utilizzato perché l'utente non è in prossimità di esso e lo riattiva poco prima della richiesta di servizio dell'utente. Questo è possibile grazie ad un meccanismo di pre-wake-up che considera la prossimità dell'utente come sua intenzione di utilizzare il PC. Per rilevare la posizione relativa dell'utente rispetto al PC viene utilizzato il sistema Bluetooth del cellulare che si suppone sia nella tasca dell'utente. Infine, Christensen et al. hanno realizzato la soluzione [61] in [62], utilizzando un piccolo dispositivo wireless USB per bloccare il PC quando l'utente si allontana dalla postazione modificandolo in modo da porre il PC in stato sleeping, e per risvegliarlo quando l'utente si avvicina nuovamente.

2.1.3.4 Protocolli e Applicazioni Power-Aware

Infine, illustriamo brevemente le tecniche per la (iv) progettazione di protocolli e applicazioni power-aware offerte dalla letteratura. Esistono tecniche relative alla modifica dei protocolli di rete in modo che diventino energy-aware e tecniche relative al livello applicazione. Relativamente ai protocolli di rete, per risolvere il problema del mantenimento della connessione TCP quando un host passa in stato sleeping senza chiudere le connessioni TCP attive, sono state proposte due soluzioni: il protocollo Green TCP/IP [63], che è una versione del protocollo TCP/IP modificata, e il meccanismo di split TCP connection [12], basato sull'aggiunta di un livello applicazione che gestisca le connessioni TCP.

Green TCP/IP proposto in [63] aggiunge al protocollo TCP legacy il concetto di connessione in stato sleeping. Il client Green TCP/IP notifica che sta andando in stato sleeping ad un Green TCP/IP server, il quale logicamente mantiene la connessione attiva, ma non invia alcun dato o pacchetti ACK al client dormiente. Quando il client Green TCP/IP si risveglia lo notifica al server ed il flusso di dati verso esso viene immediatamente ripristinato. Le modifiche al TCP/IP per implementare la connessione del PC dormiente devono essere compatibili con la versione legacy in modo che gli host Green TCP/IP possano coesistere con quelli non-Green, o normali. Per questa ragione è stata creata l'opzione TCP-SLEEP nell'header TCP, che serve a notificare al server che il client sta andando in stato sleeping.

Un altro approccio, che propone una soluzione implementata a livello applicazione, è lo split della connessione TCP aggiungendo un livello *shim* tra il socket ed il livello application [12]. Quando il client va in stato sleeping, il livello shim lo notifica al livello shim dell'altro capo della connessione per chiuderla. Durante il periodo di sleeping, se il server vuole inviare dati al client, prima lo deve risvegliare e ristabilire la connessione TCP. Relativamente alle applicazioni, quelle esistenti, come ad esempio le popolari piattaforme per il P2P Internet file sharing (come BitTorrent) o i programmi VoIP (come Skype) sono state realizzate col presupposto che il PC utente rimanga sempre attivo. Per essere effettivamente utilizzate con PC dormienti che possono essere svegliati con meccanismi di on-demand wake-up, essi devono essere appropriatamente adattati. Recenti studi [11] indicano che BitTorrent è il protocollo P2P più popolare (circa il 50 % - 75 % dell'intero

traffico P2P). Anche la nostra soluzione si inserisce in questa sezione, come un'applicazione che viene resa energy friendly quando di base non lo è.

Green BitTorrent [64] è una versione modificata del popolare protocollo BitTorrent che permette ai nodi che hanno completato il processo di download, e che non sono momentaneamente coinvolti in alcuna operazione di upload, di andare in stato sleeping. Dal punto di vista di un generico nodo, i nodi che scaricano lo stesso file possono essere in uno dei seguenti stati: *connected*, se la connessione TCP è attiva, *sleeping*, se il nodo è disconnesso, ma la connessione TCP potrebbe essere ristabilita, è *unknown*. Quando il nodo considerato rileva la disconnessione di un altro nodo, lo stato di quest'ultimo nella lista del nodo preso in esame è settato a sleeping, ma la connessione TCP non viene chiusa. Quando il numero di nodi è inferiore ad una soglia predefinita, il nodo considerato può esplicitamente svegliare un nodo inviandogli uno speciale messaggio di wake-up. Il nodo svegliato viene selezionato in modo casuale da una lista fornita dal gestore delle connessioni BitTorrent e viene stabilita una connessione TCP. Lo stato del nodo svegliato nella lista del nodo considerato è settato ad unknown. Green BitTorrent è compatibile con la versione legacy di BitTorrent, anche se il secondo sperimenta un degrado delle performance poiché chiude tutte le connessioni con nodi che vanno in stato sleeping. Ma Green BitTorrent, con una semplice modifica a BitTorrent legacy e un piccolo incremento del tempo di download, consente di ottenere fino al 75 % di risparmio energetico.

2.2 *Discussione*

In questo capitolo abbiamo analizzato le principali tecniche proposte per aumentare l'efficienza energetica di Internet. Abbiamo classificato in particolare le soluzioni proposte in letteratura sulla base di una tassonomia che copre sia le soluzioni maggiormente studiate che, per completezza, approcci diversi.

Per ridurre il consumo energetico delle reti, è necessario utilizzare tecniche di power management. Dato che il power management tradizionale non può essere utilizzato in equipaggiamenti e dispositivi di rete, è stato ideato il nuovo concetto di Dynamic Power Management (DPM). L'idea è l'energy proportional computing, caratterizzato da differenti livelli di utilizzo di potenza in funzione del livello di utilizzazione. DPM si basa sulla predizione dei periodi di inattività perché durante questi periodi, gli elementi connessi in

rete o i loro componenti possono essere messi in stato sleeping.

La maggior parte del consumo diretto di energia si concentra nella rete di accesso. Abbiamo illustrato tecniche di energy saving per link, dispositivi di rete e protocolli e applicazioni.

Nella categoria di spreco energetico relativo ai *link*, la velocità del collegamento gioca un ruolo molto importante, dato che a velocità elevate corrispondono elevati consumi. La tecnica maggiormente studiata è l'adattamento della velocità del link alla sua utilizzazione. Questo metodo, che sembra essere molto efficace per l'efficienza energetica dei link (ALR, RPS, LPI), è stato scelto dallo standard IEEE 802.3az Energy Efficient Ethernet (EEE). Questa è una soluzione molto generale che può essere adattata ad ogni tipo di link. Tutte le tecniche per l'efficienza energetica proposte per i link non introducono nella rete un alto livello di packet loss per evitare una riduzione della Quality of Service percepita dall'utente. Infatti la sfida è il compromesso tra packet loss e, in generale QoS, e risparmio energetico.

Nell'area dei *dispositivi di rete*, data la capacità di molte interfacce di commutare tra differenti modalità di consumo di potenza, tecniche di sleeping sono impiegate per risparmiare energia. È possibile mettere in stato sleeping componenti di dispositivi [22, 27] o interi sistemi [28]. Per decidere quando e per quanto tempo restare inattivo, il dispositivo può basarsi sulla previsione del prossimo istante di arrivo di dati o sul numero di pacchetti che possono arrivare nel periodo di inattività, o può modellare il traffico regolando il flusso per consentire lo stato di sleeping. In ogni caso possono verificarsi perdita di pacchetti e ritardo. La perdita di pacchetti non è influenzata dal carico medio ma dal traffico a blocchi. Il ritardo è influenzato dallo stato di sleeping e in [28] dipende dalla dimensione del buffer. Per risolvere il problema energetico nell'ambito di *protocolli e applicazioni* è necessario un nuovo approccio power-aware nella loro progettazione. L'attuale obiettivo della progettazione di rete è l'alta affidabilità e per questo motivo le reti sono densamente interconnesse con conseguente spreco energetico. Il nuovo metodo progettuale power-aware dovrebbe includere il compromesso tra prestazioni e risparmio energetico, meccanismi per mettere in stato sleeping componenti e nodi, e la capacità di rispondere dinamicamente ai cambiamenti della rete. [31] e [32] suggeriscono algoritmi di ottimizzazione per lo spegnimento di nodi e collegamenti per cambiare la topologia di rete allo scopo di minimizzare il consumo di energia. Un altro approccio, a livello di routing,

viene proposto da [33] e [34] per instradare il traffico in modo efficiente dal punto di vista energetico. In tutte queste soluzioni, un aspetto molto importante è l'approccio energy-aware nella progettazione ad ogni livello: a livello di componenti, a livello di dispositivi e a livello di rete. È inoltre molto importante l'impiego di componenti ad alta efficienza energetica.

D'altra parte abbiamo considerato il consumo di energia indotto che è causato dai data center e dagli edge device. Questo è molto maggiore del consumo di energia diretto perché i server e gli edge device sono molto numerosi e diffusi.

Per i *data center* sono stati presentati due approcci per ottenere alta efficienza energetica: il bilanciamento di potenza nei cluster di server in funzione dei carichi di richiesta, e la progettazione energy proportional dei server sviluppando macchine che consumano energia in proporzione alla quantità di lavoro svolto. Come sottolineato in [35], l'approccio [36] presenta alcuni limiti in quanto, specialmente per i servizi su larga scala, il carico (dati e operazioni) è distribuito su tutte le macchine per prevenire la perdita dei dati e per i requisiti di fault tolerance. Inoltre, spesso i server eseguono operazioni in background e non possono entrare in stato di sospensione. Per queste ragioni, è difficile spegnere completamente un sottoinsieme di server in un cluster, tutti i server dovrebbero essere disponibili, anche in periodi di bassa utilizzazione. Inoltre, a basso carico la commutazione di server consapevole dal punto di vista energetico proposto in [36] sperimenta elevata latenza di servizio a causa del numero di server spenti. Dall'altra parte, mentre [35] introduce una nuova metodologia di progettazione suggerendola per tutti i componenti del server, la soluzione proposta in [36] non necessita di convincere i progettisti dell'importanza della questione energetica e di un nuovo approccio. Per questa ragione, l'idea in [36] sembra essere la soluzione più semplice da impiegare nel prossimo futuro, mentre [35] punta ad una rivoluzione a lungo termine lungimirante e consapevole energeticamente nella progettazione di componenti server e sistemi futuri, dove l'energy proportional diventerà un obiettivo primario di design.

Gli *edge device* sono molto diffusi sia in ambito domestico che aziendale e probabilmente rappresentano l'ambito più vasto in cui ottenere il risparmio energetico delle reti. Tecniche di proxying e wake-up sono le soluzioni maggiormente studiate per l'efficienza energetica negli edge device, anche accoppiati. Un proxy potrebbe permettere agli host di entrare in

stato sleeping e, mentre questi sono inattivi, potrebbe mantenere la loro presenza in rete. Meccanismi di wake-up sono necessari per risvegliare gli host quando i requisiti di rete richiedono la loro presenza attiva. Un proxy è un'entità a basso consumo che può essere posizionato a livello di NIC, integrato nell'host dormiente, a livello LAN, posizionato nella stessa LAN dell'host, o a livello di rete Internet. Quindi può essere interno oppure esterno all'host. La soluzione interna permette affidabilità del proxy anche in configurazioni complesse ed un'ottima coordinazione tra host e proxy perché è customizzato per lo specifico PC. D'altra parte, quello esterno potrebbe utilizzare dispositivi di rete pre-esistenti e potrebbe servire un maggior numero di host, riducendo molto il suo impatto energetico. La maggior potenzialità del proxy è la sua capacità di risolvere problemi legati all'incompatibilità tra connettività e power management negli host, introducendo per questi dispositivi un elevato risparmio energetico potenziale. Molte applicazioni proxy sono state presentate in letteratura e anche in prodotti commerciali. Alcuni esempi sono Somniloqui [54], NCP per reti Ethernet [52] e, generalizzato per reti IP [12, 15, 19], implementato per protocolli specifici [46, 65], associato a meccanismi WoL [64], o implementato in prodotti commerciali [48]. Le soluzioni proxy illustrate sono implementate ad ogni livello, dai protocolli di base (ad esempio ARP, DHCP, TCP SYN), ai protocolli di alto livello (ad esempio SIP), alle applicazioni (ad esempio P2P), fino a intere immagini di PC (utilizzando la virtualizzazione).

Per applicazioni che non sono compatibili col proxying, si può sperare che futuri aggiornamenti vadano nella direzione di compatibilità. Soluzioni di risparmio energetico in relazione al protocollo TCP/IP necessitano di modifiche dello standard e la compatibilità all'indietro risulta quindi necessaria per farne delle proposte realmente utilizzabili.

Tra le soluzioni dipendenti dal contesto la letteratura propone meccanismi per spegnere periferiche, dispositivi e interi sistemi. Queste applicazioni utilizzano approcci diversi: fanno uso di reti per rilevare informazioni di contesto, in particolare relativamente a utenti e alla loro posizione, per prendere decisioni di power management.

In generale risulta molto complicato cambiare radicalmente protocolli, metodi e meccanismi del core di Internet, principalmente per garantire il fondamentale principio della compatibilità all'indietro. Quindi, le maggiori opportunità per apportare cambiamenti sono rappresentate dall'ambito degli edge device, in particolare a livello di progettazione di

applicazioni e protocolli, permettendo a host e equipaggiamenti di rete di essere disattivati. Siamo convinti che questa classe di approcci otterrà grande importanza e attenzioni nella ricerca in futuro. Per questo, anche il nostro settore di ricerca, presentato in questa tesi, si inserisce proprio nell'ambito degli edge device, con una soluzione proxy-based che consenta agli host di andare in stato sleeping per risparmiare energia, ed in particolare nell'ambito delle applicazioni P2P che non sono pensate originariamente in modo energy friendly.

Capitolo 3

Protocollo BitTorrent

Le Peer-to-Peer (P2P) overlay network sono reti distribuite ed autorganizzanti situate sulla rete IP. I nodi appartenenti alla rete sono detti *peer*, cioè nodi equivalenti che fungono sia da client che da server verso altri nodi della rete. Questo modello di rete è in antitesi con l'architettura gerarchica client-server dal momento che introduce simmetria e fornisce un buon substrato per la condivisione di file su vasta scala, distribuzione di contenuti e applicazioni multicast.

Le P2P overlay network si possono classificare in due classi: quelle *non strutturate*, in cui i peer diventano parte della rete seguendo semplici regole, non sfruttano conoscenze a priori sulla topologia di rete e per effettuare richieste sfruttano il meccanismo del flooding fino a trovare il contenuto cercato. E quelle *strutturate*, con una topologia controllata, in cui i contenuti sono posti in locazioni specifiche tali da rendere efficiente il meccanismo di ricerca, e che spesso sfruttano come substrato una Distributed Hash Table (DHT) per mappare la coppia informazione-nodo in cui è locata.

Le reti P2P strutturate sono più efficienti, soprattutto per la ricerca di contenuti rari, ma comportano un maggior overhead, mentre quelle non strutturate sono reti naturalmente ad hoc e quindi non unificabili sotto una comune piattaforma per sviluppi applicativi, ma comportano un minor overhead, e per questo sono il tipo di P2P overlay network più diffuso [66].

3.1 Descrizione del protocollo

BitTorrent è una overlay network di tipo non strutturato per la condivisione e la distribuzione di file di grandi dimensioni attraverso Internet [67]. BitTorrent, termine che si riferisce sia alla rete P2P che al protocollo che al client originario, fu sviluppato da Bram Cohen nel 2002 e scritto in linguaggio Python.

L'obiettivo di BitTorrent è realizzare e fornire un sistema di file sharing efficiente, ridistribuendo il carico dell'upload su tutti i computer che concorrono al download del file; infatti, una volta che un utente ha parte del file, inizia esso stesso a fornire quella parte agli altri utenti, alleggerendo il carico del detentore del file completo e permettendo così un possibile totale sfruttamento della banda di download.

BitTorrent risulta essere così un meccanismo per coordinare il lavoro di molti computer ottenendo la più elevata velocità di download possibile.

I nodi che costituiscono la overlay sono chiamati in generale *peer*, e si distinguono in *seed*, cioè i detentori di una intera copia del file condiviso, e *leecher*, che possiedono invece nessuna o solo alcune parti del file. I file condivisi sono anche chiamati *torrent* e i peer coinvolti nella distribuzione di uno specifico torrent costituiscono lo *swarm*. Il file è suddiviso in piccole parti della stessa dimensione (tipicamente 256 kB) chiamate *chunk*, che rappresentano l'unità di condivisione o di scambio tra peer, e che ricomposte poi a destinazione ricostituiranno l'intero file. Ad ogni chunk è associata una fingerprint ottenuta con l'algoritmo SHA-1 per garantirne l'integrità. Ogni chunk è a sua volta suddiviso in sottoparti della stessa dimensione (tipicamente 16 kB) dette *blocchi*.

L'idea di base di BitTorrent è il fatto che i peer contemporaneamente scaricano e forniscono chunk ad altri peer. Questo è reso possibile dal fatto che ogni peer scarica un certo file da una moltitudine di altri peer (architettura P2P) invece che da un singolo server come nella tradizionale architettura client-server. Il modello P2P consente, con tale tipo di condivisione cooperativa, di ottenere una capacità del processo di download maggiore rispetto a quella ottenuta tradizionalmente scaricando da un singolo server [68].

Come mostrato in Figura 4, un peer che desidera scaricare un file deve procurarsi il corrispondente *file.torrent*. Esso è un file di piccole dimensioni, tipicamente residente su un Web server (*torrent server*), e può essere trovato tramite un normale motore di ricerca. Un *file.torrent* contiene una serie di informazioni relative al file, in codifica BenCode, tra cui il nome del file, la dimensione, le parti in cui è suddiviso e la fingerprint SHA-1 di ogni parte, e il nome del *tracker*. Quest'ultimo è un nodo che tiene traccia costantemente di quali peer hanno quali chunk del file. Quando un peer entra a far parte di uno swarm esso si registra con il tracker e, poi, periodicamente informa il tracker della sua presenza nello swarm.

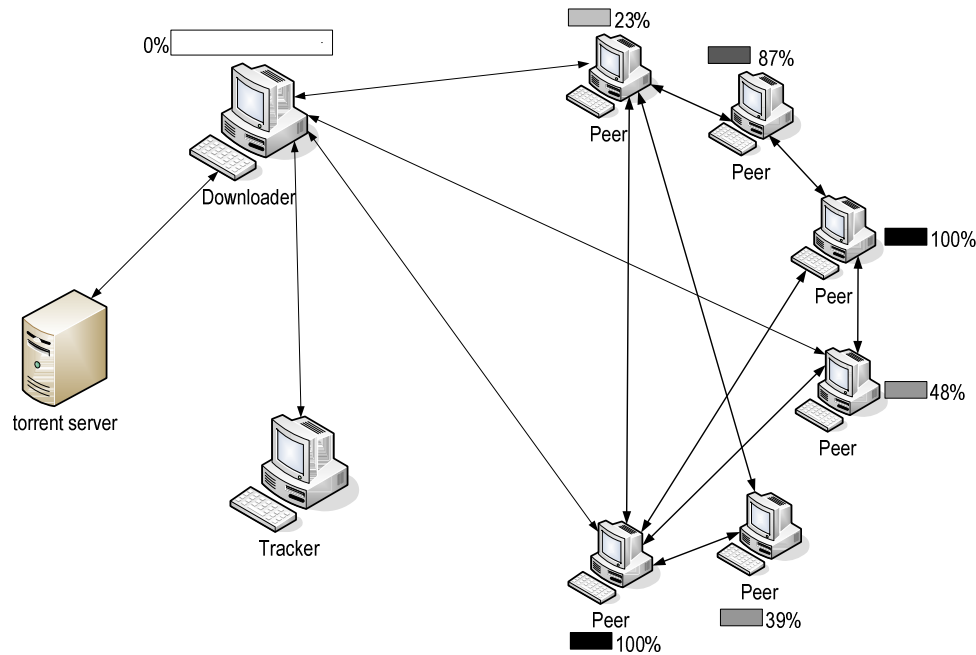


Figura 4 - Funzionamento ad alto livello del protocollo BitTorrent.

Una volta ottenuto l'indirizzo del tracker, scaricando il file.torrent da un torrent server, il peer che desidera scaricare il file apre una connessione TCP/IP con esso e riceve una lista di peer da poter contattare per iniziare il processo di download. In ogni istante il peer sarà connesso con un set di peer 'vicini', chiamati appunto *neighbor*, coi quali si scambierà parti del file. L'insieme dei *neighbor* varia dinamicamente dal momento che, periodicamente, alcuni peer lasciano lo swarm, mentre altri entrano a farne parte, ma se il numero di *neighbor* scende al di sotto di una certa soglia (tipicamente 20), il nodo richiede al tracker un'ulteriore lista di peer da poter contattare. Inoltre, ogni peer seleziona i vicini che preferisce per scaricare i chunk, e sceglie in particolare quei peer da cui ottiene la velocità di download più elevata. Inoltre, ogni 30 secondi altri vicini sono selezionati in modo casuale, come modo per scoprire nuovi *neighbor* e permettere a nuovi peer di prendere parte allo swarm.

In un certo istante, ogni peer dello swarm avrà un sottoinsieme dei chunk del file. Per capire da quale vicino scaricare i chunk mancanti, il peer considerato chiede a ciascuno dei suoi vicini la lista dei chunk che possiede. Per decidere quale chunk richiedere prima il

nodo considerato utilizza la *Rarest First* policy, meccanismo attraverso il quale esso attribuisce maggior priorità a quei chunk che sono meno diffusi nello swarm. Specificamente, ogni peer mantiene il numero di copie di ogni parte contenuta nel suo vicinato (numero di peer dello swarm che hanno quella parte del file) e definisce il *rarest pieces set*, l'insieme delle parti meno diffuse tra i suoi vicini (quelle che compaiono in numero minore). Tale insieme è aggiornato ogni volta che una copia di una parte viene aggiunta o tolta e ogni peer seleziona random una parte tra quelle del *rarest pieces set* per farne il download. L'obiettivo di tale politica è massimizzare la diversità dei contenuti del sistema ed assicurare una distribuzione uniforme delle parti tra i nodi. Questo rende il sistema un po' più lento a causa del fatto che le parti 'rare' sono difficili da reperire.

Tale comportamento può essere modificato con 3 politiche aggiuntive.

La *Random First Policy*, adottata per la scelta delle prime parti da scaricare. Un nuovo nodo che ha scaricato da 0 a 4 parti del file non adotta la politica *Rarest First*. Dato che per il nuovo nodo è importante effettuare un bootstrap veloce per partecipare alla rete P2P più velocemente possibile, utilizza la prima opportunità di download disponibile, scaricando random le prime 4 parti.

Essendo inoltre ogni chunk a sua volta suddiviso in blocchi, entra in gioco anche la *Strict Priority Policy*. Questa politica agisce a livello di blocco e rende più veloce il completamento di un intero chunk. Quando almeno un blocco è stato richiesto, anche gli altri blocchi dello stesso chunk vengono richiesti con la più alta priorità, in modo da completare velocemente intere parti che sono le unità di condivisione del file (mentre i singoli blocchi non lo sono).

Infine, dato che la fine di un download può essere potenzialmente ritardata a causa di un peer molto lento in download, il meccanismo *Endgame Mode* consente di evitare questo rallentamento. Individuati dal peer considerato tutti i blocchi mancanti per completare il download del file, esso manda in broadcast a tutti i suoi vicini la richiesta. Quando uno di essi risponde, il client avvisa tutti gli altri che la richiesta per quel blocco è scaduta così da risparmiare banda ed evitare duplici download dello stesso blocco.

Per decidere a quali richieste provenienti da altri peer rispondere, il nodo considerato si basa sulla *Tit-for-Tat* (TAT) policy, cioè attribuisce maggior priorità a quei nodi da cui scaricano dati a velocità superiore. Specificamente, per ciascuno dei suoi neighbor il nodo

considerato misura la velocità di download e poi seleziona i quattro peer che gli stanno fornendo dati alla velocità maggiore per ricambiare in modo equo. Questo meccanismo consente ai peer di difendersi dai free-riding, utenti che intendono scaricare alla massima velocità per poi, appena terminato, disconnettersi dalla rete. Per essi, la velocità di download da altri peer viene ridotta se questi limitano la loro velocità di upload.

L'obiettivo di ogni client è quello di massimizzare il suo download rate e per farlo utilizza l'*algoritmo di choking*. Il choking, soffocamento, è il meccanismo usato per limitare il numero di upload concorrenti, cioè il rifiuto temporaneo di fare upload ad un vicino, mentre il download può continuare e la connessione non deve essere rinegoziata quando il soffocamento termina. Soltanto le connessioni ad un numero massimo di vicini (tipicamente 4) sono unchoked, cioè sbloccate, ed è concesso l'upload. Un nodo rivaluta il rate di download che sta ricevendo dai vicini ogni 10 secondi per decidere quale nodo, attualmente unchoked, può essere choked ed essere rimpiazzato da un altro vicino. L'insieme di nodi ai quali un nodo sta facendo upload è detto unchoke set. Generalmente tale insieme non coincide con quello dei vicini da cui sta facendo download.

L'*Optimistic Unchoking* è una politica che permette di fare unchoking su un nodo scelto random e non in base al suo download rate; tale meccanismo è utilizzato periodicamente, ogni 30 secondi ed ha un duplice scopo: consentire ad un nodo di scoprire nuovi vicini che potrebbero offrire un download rate superiore rispetto al download rate dei nodi da cui fa correntemente il download e consentire ad un nuovo nodo, che non ha niente da offrire, di scaricare il suo primo blocco.

Può accadere che un client venga soffocato da tutti i peer da cui sta scaricando e che esso continui a fornire dati in upload. Questo evento causerebbe una drastica diminuzione, se non un annullamento, del suo download rate. Per evitare ciò un client che non riceve parti di un file per più di un minuto soffoca subito la connessione, assumendo di essere stato snobbato dal peer in questione, e dà il via ad un optimistic unchoking. Potrebbe continuare a cooperare con il peer snob, solo se il peer risultasse essere il bersaglio dell'optimistic unchoking.

Una volta che il client ha terminato il download del file diventa da leecher un seed: il cooperatore per eccellenza. A questo punto però il client non ha più i suoi valori di download rate su cui basare le sue scelte, quindi sceglie di fare upload ai peer 'più leecher',

cioè quelli con meno parti del file; questo principio potrebbe essere chiamato il principio di 'leechest first'.

Ulteriori dettagli sul protocollo BitTorrent sono riportati nelle specifiche in Appendice A.

3.2 Funzionamento del protocollo

Il funzionamento del protocollo BitTorrent, espresso in termini delle varie azioni che gli attori compiono, è illustrato in Figura 5.

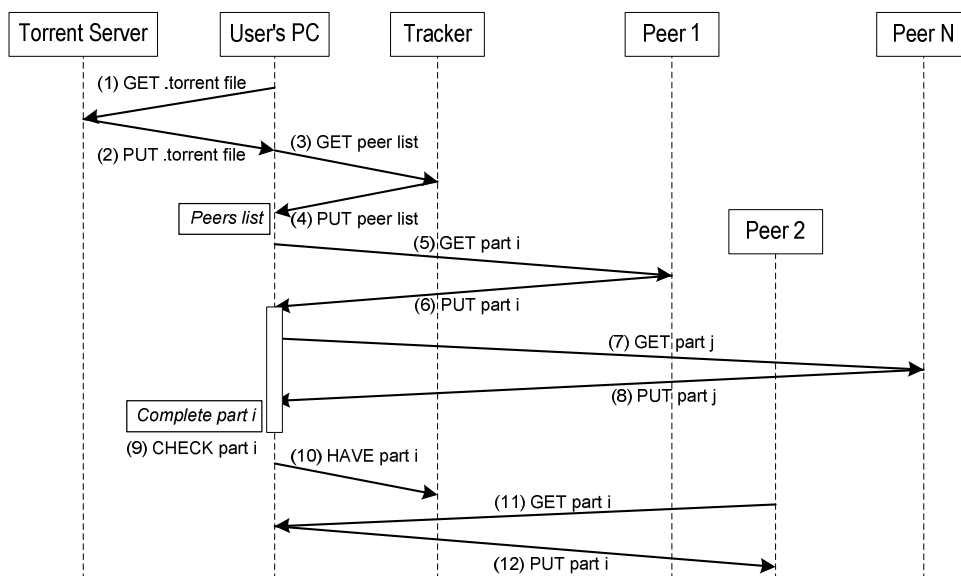


Figura 5 - Il protocollo BitTorrent.

L'utente, tramite il browser, si collega ad un comune motore di ricerca ed effettua una ricerca per il file.torrent desiderato (che ha lo stesso nome del file che l'utente vuole scaricare seguito dall'estensione .torrent). Tra i risultati forniti dal motore di ricerca l'utente ne sceglie uno e viene redirezionato sulla pagina web di un torrent server. Questo fornisce all'utente una lista dei torrent disponibili relativi al file desiderato. L'utente, tra le varie opzioni disponibili, sceglie un torrent con un elevato numero di seed e leecher e scarica il file.torrent relativo ad esso (fasi 1-2 in Figura 5). Cliccando sul file.torrent salvato si avvia

il client BitTorrent installato sul PC dell'utente che si connette al tracker, il cui indirizzo è contenuto all'interno del file stesso. Il tracker fornisce al client una lista di peer (un sottoinsieme dello swarm) che hanno tutto o parte del file (fasi 3-4). Il client BitTorrent residente sulla macchina dell'utente si connette quindi ai peer fornitigli dal tracker ed inizia il download di parti del file (fasi 5-6-7-8). Quando il client BitTorrent ha completato il download di una parte, ne effettua il controllo con l'algoritmo SHA-1 (fase 9) e, se il controllo va a buon fine, diventa esso stesso fornitore di quella parte del file. Per far questo, informa il tracker che detiene una parte di file completa (fase 10) in modo che quest'ultimo possa fornire il suo indirizzo IP ad altri peer dello swarm che desiderano scaricare il file (fasi 11-12). Il client BitTorrent informa il tracker delle parti complete di cui è in possesso ogni 30 minuti.

Nel caso in cui il numero di peer a cui il client BitTorrent è connesso scenda al di sotto di una certa soglia (tipicamente 20), il client effettua una richiesta al tracker di nuovi indirizzi IP di peer a cui connettersi.

3.3 *Discussione*

BitTorrent è un protocollo non efficiente dal punto di vista energetico. Esso prevede che i peer restino connessi alla overlay network durante l'intero processo di download del file che desiderano scaricare, che impiega tipicamente alcune ore. Ne consegue un elevato consumo energetico.

Spegnere periodicamente i peer non è possibile per vari motivi. In primo luogo, se un peer sta scaricando un file, spegnerlo non introduce alcun risparmio energetico in relazione al download in corso in quanto il download si ferma appena il peer viene spento. Inoltre, spegnere i peer che non stanno scaricando nessuna parte del file non rappresenta una soluzione efficiente in quanto ne potrebbe conseguire una riduzione delle prestazioni a livello di swarm. Infine, pensare ad uno spegnimento coordinato dei peer richiederebbe un meccanismo centralizzato che quindi verrebbe meno alla natura del paradigma P2P che sta alla base di BitTorrent.

Il problema risiede nel fatto che BitTorrent si basa sul mantenimento di connessioni TCP tra peer. Quando un peer avverte che un suo vicino si è disconnesso o non è raggiungibile, tale nodo viene eliminato dalla sua lista di neighbor perché è considerato 'dead'. Un peer

spento non può mantenere connessioni TCP con altri peer o rispondere a richieste di connessione TCP, quindi non può prendere parte al file sharing tramite BitTorrent.

Per superare le limitazione di questo protocollo dal punto di vista energetico ne abbiamo progettato, ideato e valutato sperimentalmente una versione proxy-based.

Capitolo 4

Eenergy Efficient BitTorrent

In questo capitolo illustriamo la nostra proposta Energy Efficient BitTorrent (EE-BitTorrent [69]) per introdurre efficienza energetica in applicazioni P2P che sono responsabili di un elevato consumo energetico nei personal computer. In particolare, la nostra soluzione è customizzata per il protocollo BitTorrent e si inserisce, dal punto di vista architetturale, nell'ambito delle soluzioni proxying per gli edge device. EE-BitTorrent è stato valutato sperimentalmente sia in ambito aziendale / dipartimentale, con un elevato numero di PC connessi ad una LAN, sia in ambito residenziale / domestico in cui tipicamente un solo PC esegue l'applicazione P2P.

4.1 Motivazioni

Come già detto, l'uso di BitTorrent è molto diffuso e contribuisce quindi significativamente al consumo energetico sui PC, che vengono lasciati accesi dagli utenti per scaricare i contenuti desiderati. Si presenta perciò il problema di cercare un compromesso tra le esigenze dell'utente ed il risparmio energetico. L'obiettivo è quello di ottenere risparmio energetico per motivi economici e ambientali, pur consentendo agli utenti di condividere file, senza comprometterne l'attività o ridurre la QoS da essi percepita per l'attività di file sharing. Gli utenti vogliono utilizzare BitTorrent come sistema di condivisione di file, vogliono sfruttare i periodi in cui non sono presenti alla loro postazione, come la notte ed il weekend, per scaricare file e non vogliono subire l'impatto dell'uso di tecniche di risparmio energetico. Il punto sta proprio nel soddisfare i bisogni dell'utente e contemporaneamente ridurre i consumi di potenza in modo trasparente utilizzando tecniche di efficienza energetica.

Energy Efficient BitTorrent (EE-BT) è una versione proxy-based di BitTorrent la cui architettura è mostrata in Figura 6. I componenti architetturali sono un certo numero di PC di utenti che utilizzano client BitTorrent e che nel momento in cui partecipano alla overlay

network diventano *BitTorrent Peer*, ed un PC posto tra i peer ed il resto della rete BitTorrent che assume il ruolo di *BitTorrent Proxy*. Il proxy può essere una macchina dedicata a tale ruolo, oppure una macchina che deve rimanere costantemente accesa per fornire altri servizi (ad esempio DHCP, Web proxy, ecc.). Chiaramente, dal punto di vista del risparmio energetico, il secondo caso è preferibile.

I peer posti 'dietro' il proxy BitTorrent fanno richiesta al proxy stesso di scaricare il contenuto desiderato per conto loro. Il proxy partecipa alla overlay BitTorrent nel modo convenzionale e si fa carico dei download di tutti i peer. Mentre i download sono in corso, i peer dietro al proxy possono essere spenti senza fermare il download del file desiderato. Infine, i file richiesti vengono trasferiti dal proxy ai peer dopo il completamento.

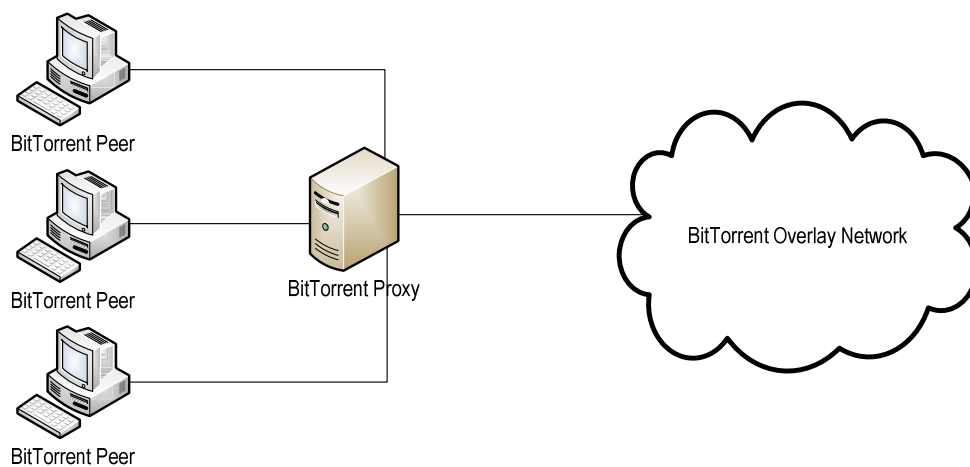


Figura 6 - Rappresentazione ad alto livello dell'architettura Energy Efficient BitTorrent.

Questa architettura si presta chiaramente al risparmio energetico, e mantiene anche i principi P2P su cui si fonda l'architettura BitTorrent originale. L'intera rete BitTorrent non viene assolutamente modificata dal momento che il proxy agisce esattamente come uno standard BitTorrent peer. Le uniche modifiche richieste sono apportate al proxy e ai PC utente posti dietro ad esso. Infatti, proxy differenti sono completamente indipendenti. Pertanto, questa architettura risulta essere anche scalabile, in quanto non richiede modifiche all'architettura BitTorrent globale, né un coordinamento globale tra (insiemi di) peer BitTorrent. Infine, questa architettura è adatta anche per supportare i client mobili che

accedono a Internet, ad esempio, attraverso punti di accesso Wi-Fi connessi alla rete in cui il proxy è in esecuzione, e, più in generale, è una soluzione che consente il download asincrono tramite BitTorrent, che è un aspetto non supportato dall'architettura BitTorrent convenzionale.

4.2 Definizione del Protocollo EE-BitTorrent

L'architettura proposta rientra nella famiglia delle tradizionali architetture a blocchi, ad esempio, [43]. Le componenti architetturali tra un peer e il proxy sono mostrati in Figura 7. BT Peer sul proxy è un peer BitTorrent standard. Questo peer si occupa di scaricare i contenuti richiesti da tutti gli utenti dietro il proxy.

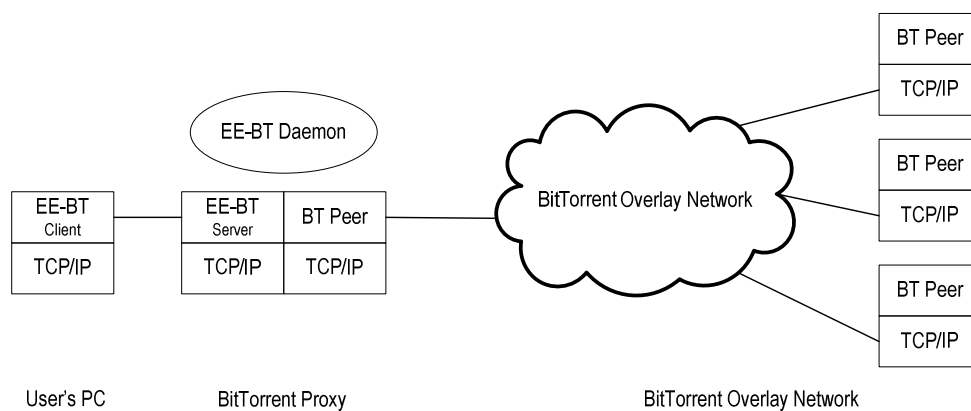


Figura 7 - Architettura di Energy Efficient BitTorrent.

Nella parte 'interna' dell'architettura (cioè tra il proxy BitTorrent e il PC dell'utente), si adotta un semplice schema client-server, implementato dal modulo EE-BT Client sul PC dell'utente che funge da client e il modulo EE-BT Server sul proxy che funge da server. Il server EE-BT controlla costantemente le richieste in arrivo per i nuovi download provenienti da uno dei vari client EE-BT dietro esso, che vengono passate al EE-BT Daemon. Poi, il modulo EE-BT Daemon traduce queste richieste in richieste di download prese in carico dal BT Peer residente sul proxy connesso alla rete BitTorrent.

Oltre alle richieste per il download di nuovi file, un client BT può anche emettere comandi per conoscere lo stato dei file precedentemente richiesti, così come comandi per prelevare i

file richiesti dal proxy, una volta che sono stati completamente scaricati. Tra due successive richieste, il PC dell'utente può essere spento (o messo in stand-by). Gli utenti BitTorrent possono anche caricare sul proxy BitTorrent i contenuti che desiderano condividere sulla overlay BitTorrent. Questo è un vantaggio ulteriore e importante della nostra architettura. Il BT Peer sul proxy può condividere tutti i file che sarebbero condivisi dai singoli client in esecuzione sui PC degli utenti. Pertanto, il BT Peer sul proxy ha la possibilità di scaricare con una banda superiore a quella di qualsiasi altro peer che scarica individualmente (nel caso in cui nessun proxy sia utilizzato). Così, la nostra architettura proxy-based si prevede raggiunga tempi di download più bassi per tutti gli utenti, oltre a fornire un notevole risparmio energetico.

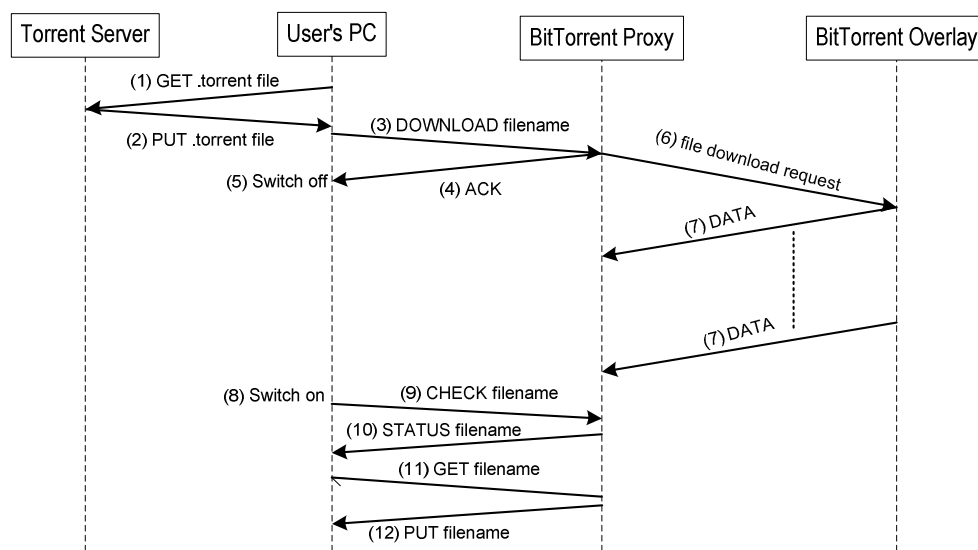


Figura 8 - Protocollo Energy Efficient BitTorrent.

L'architettura proposta richiede protocolli di rete molto semplici. La Figura 8 mostra le azioni eseguite dai vari attori durante le diverse fasi di un download di file. Quando un utente vuole scaricare un nuovo file, l'EE-BT Client in esecuzione sul PC dell'utente recupera il file.torrent da un torrent server in Internet, come nella tradizionale architettura BitTorrent (fasi 1-2 nella Figura 8). Poi, carica il file.torrent sull'EE-BT Server sul proxy

BitTorrent richiedendo il download del file desiderato (fase 3). L'EE-BT Server riconosce la richiesta ricevuta (fase 4) e passa il file.torrent al suo BT Peer attraverso l'EE-BT Daemon per avviare le operazioni di download secondo il protocollo BitTorrent standard (fasi 6-7). Dopo aver ricevuto conferma dall'EE-BT Server, notifica che il download è iniziato al EE-BT Client che a sua volta informa l'utente che la richiesta di download è stata correttamente emessa e che il processo di download è in corso. L'utente può quindi spegnere il proprio PC (fase 5). Non appena l'EE-BT Client sul PC dell'utente viene riavviato (fase 8), esso controlla lo stato di tutti i file di cui ha precedentemente richiesto il download all'EE-BT Server. Per ciascuno di essi, l'EE-BT Client chiede all'EE-BT Server un aggiornamento di stato (fasi 9-10). Se il download è terminato, l'EE-BT Client preleva il file corrispondente dal proxy (fasi 11-12).

4.3 Valutazione sperimentale in uno scenario dipartimentale

Lo scenario considerato in prima istanza è un tipico ambiente dipartimentale o aziendale, in cui è presente una LAN standard sulla quale un certo numero di utenti utilizza client BitTorrent sul proprio PC. Un computer della LAN assume il ruolo di proxy posto tra i peer (PC della LAN) ed il resto della rete BitTorrent. Quindi i peer sono posti sulla stessa LAN del proxy, che è tipicamente una rete ad alta velocità sia per quanto riguarda i collegamenti locali (tra PC utente e proxy) sia relativamente alla connessione a Internet (tra proxy e overlay BitTorrent).

In questo contesto, le modifiche necessarie si apportano sui peer BitTorrent che sono quindi confinati all'interno della sola LAN. Inoltre, il trasferimento finale del file dal proxy al PC utente richiedente avviene tra due computer connessi attraverso una rete LAN, e quindi impiega molto meno tempo di un tipico download con BitTorrent.

4.3.1 Analisi dell'efficienza energetica

Per analizzare l'efficienza energetica della nostra architettura EE-BT proxy-based abbiamo considerato il *risparmio energetico assoluto* (ΔE) e il *risparmio energetico relativo* (S), definiti come il risparmio energetico assoluto e relativo ottenuti dalla nostra architettura proxy-based rispetto all'architettura tradizionale, cioè

$$\Delta E = E_L - E_P \quad (1)$$

$$S = 1 - \frac{E_P}{E_L} \quad (2)$$

dove E_L e E_P denotano rispettivamente l'energia consumata dal PC utente per scaricare lo stesso file nell'architettura tradizionale e in quella proxy-based.

Chiaramente, l'energia consumata dal PC utente in entrambe le architetture è data dal tempo totale che esso rimane acceso, moltiplicato per il consumo di potenza del P_{PC} .

Indichiamo con t_L e t_P il tempo totale che il PC utente deve rimanere acceso per scaricare completamente il file rispettivamente nell'architettura tradizionale e proxy-based.

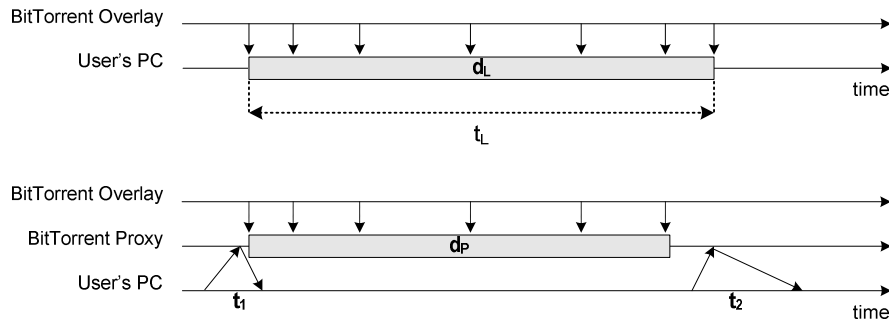


Figura 9 - Processo di download del file nell'architettura legacy (sopra) e proxy-based (sotto).

Come mostra la Figura 9, nell'architettura tradizionale questo tempo corrisponde al tempo necessario al PC utente per scaricare il file, cioè $t_L = d_L$. Nell'architettura proxy-based, oltre al tempo d_P richiesto al proxy per scaricare il file, bisogna considerare anche il tempo impiegato dal PC utente per (i) delegare il download del file al proxy (t_1) e per (ii) prelevare lo stesso file dal proxy, una volta che questo è stato scaricato (t_2). Quindi, assumendo che il proxy sia una macchina che deve rimanere continuamente accesa per altre ragioni (cioè una macchina multi-server) e potendo quindi trascurarne il consumo energetico, le equazioni (1) e (2) possono essere scritte come segue.

$$S' = 1 - \frac{t_P \cdot P_{PC}}{t_L \cdot P_{PC}} = 1 - \frac{t_1 + t_2}{t_L} \quad (3)$$

$$\Delta E' = (d_L - t_1 - t_2) \cdot P_{PC} \quad (4)$$

Invece, quando il proxy è una macchina dedicata, è necessario considerare esplicitamente il suo consumo energetico. Per chiarezza, indichiamo con S'' e $\Delta E''$ il risparmio energetico relativo e assoluto quando il consumo energetico del proxy non può essere trascurato.

$$S'' = 1 - \frac{t_1 + d_P + t_2}{d_L} \approx 1 - \frac{t_1 + t_2}{d_L} \quad (5)$$

$$\Delta E'' = (d_L - (t_1 + t_2)) \cdot P_{PC} - d_P \cdot P_P \approx -(t_1 + t_2) \cdot P_{PC} \quad (6)$$

Nell'equazione (6), P_P indica il consumo di potenza del proxy. L'ultimo passaggio in entrambe le equazioni (5) e (6) è conseguenza dell'assunzione che il proxy BitTorrent è un PC simile al PC utente e, quindi, $d_L \approx d_P$ e $P_P = P_{PC}$. Si noti che assumere $d_L \approx d_P$ è un'assunzione pessimistica perché in generale il tempo di download utilizzando l'architettura proxy-based è inferiore rispetto a quando il proxy non viene utilizzato.

Le equazioni (5) e (6) mostrano chiaramente che, con un singolo PC, il consumo energetico dell'architettura proxy-based è maggiore rispetto a quello dell'architettura tradizionale. Questo è abbastanza ovvio poiché nell'architettura proxy-based è necessario considerare il consumo energetico addizionale dovuto al proxy. Comunque, possiamo aspettarci risparmio energetico se più PC, contemporaneamente, utilizzano il proxy BitTorrent per scaricare molti file in parallelo.

Generalizziamo quindi gli indici S e ΔE al caso con n differenti utenti che scaricano un file in parallelo usando BitTorrent sul proprio PC. Indicando con $S(n)$ il risparmio energetico relativo quando il numero di PC che scaricano file in parallelo è n , le equazioni (3) e (5) possono essere generalizzate come segue:

$$S'(n) = 1 - \frac{\sum_{i=1}^n t_P(i)}{\sum_{i=1}^n t_L(i)} = 1 - \frac{\sum_{i=1}^n t_1(i) + t_2(i)}{\sum_{i=1}^n d_L(i)} \quad (7)$$

$$S''(n) = 1 - \frac{\sum_{i=1}^n t_p(i)}{\sum_{i=1}^n t_L(i)} = 1 - \frac{\sum_{i=1}^n d_p(i) + \sum_{i=1}^n t_1(i) + t_2(i)}{\sum_{i=1}^n d_L(i)} \quad (8)$$

Nell'equazione (8) $\sum_{i=1}^n d_p(i)$ indica il tempo totale che il proxy deve rimanere acceso per completare il download di tutti gli n file. Dato che gli n download sono portati avanti in parallelo, il tempo totale corrisponde al tempo massimo necessario per il download di ogni file, cioè $\sum_{i=1}^n d_p(i) = d_p^{\max}$. Quindi, l'equazione (8) può essere riscritta come

$$S''(n) = 1 - \frac{d_p^{\max} + \sum_{i=1}^n t_1(i) + t_2(i)}{\sum_{i=1}^n d_L(i)} \quad (9)$$

Infine, seguendo il medesimo approccio, possiamo derivare anche il risparmio energetico assoluto $\Delta E(n)$ come funzione del numero n di file scaricati in parallelo, quando l'energia consumata dal proxy può, oppure no, essere trascurata, cioè

$$\Delta E'(n) = \left(\sum_{i=1}^n d_L(i) - \sum_{i=1}^n [t_1(i) + t_2(i)] \right) \cdot P_{PC} \quad (10)$$

$$\Delta E''(n) = \left(\sum_{i=1}^n d_L(i) - \sum_{i=1}^n [t_1(i) + t_2(i)] \right) \cdot P_{PC} - \sum_{i=1}^n d_p(i) \cdot P_p \quad (11)$$

Dato che $\sum_{i=1}^n d_p(i) = d_p^{\max}$ e assumendo $P_p = P_{PC}$, l'equazione (11) può essere riscritta come

$$\Delta E''(n) = \left(\sum_{i=1}^n d_L(i) - \sum_{i=1}^n [t_1(i) + t_2(i)] - d_p^{\max} \right) \cdot P_{PC} \quad (12)$$

4.3.2 Risultati sperimentali

Per valutare gli indici di efficienza energetica introdotti nel paragrafo precedente abbiamo bisogno di ricavare i vari tempi durante i quali il / i PC dell'utente e il proxy devono rimanere accesi in un caso reale. A questo scopo abbiamo creato un testbed sperimentale e misurato i componenti temporali necessari per il calcolo sia di S sia di ΔE attraverso le espressioni derivate nella precedente sezione. Di seguito, dopo una breve descrizione del testbed sperimentale, discuteremo i risultati ottenuti.

4.3.2.1 Descrizione del testbed

Il testbed sperimentale è basato su un insieme di PC interconnessi da una rete Ethernet Gigabit LAN che è a sua volta collegata a Internet tramite una collegamento ad alta velocità a 100 Mbps. Sfruttando l'insieme di PC abbiamo implementato due sistemi: un sistema BitTorrent legacy e quello proxy-based che abbiamo sviluppato. Tutti i PC usano il sistema operativo Linux Ubuntu 8.04 (Hardy Heron). Il client BitTorrent (cioè il software che implementa il peer di BitTorrent) è un semplice client a linea di comando fornito con la libreria libtorrent Rasterbar.

Sfruttando i due sistemi, abbiamo effettuato una serie di esperimenti e, in particolare, abbiamo misurato il tempo di download (e dei suoi componenti) necessario per scaricare lo stesso set di file rispettivamente con l'architettura legacy e proxy-based. Più precisamente, per ogni esperimento abbiamo identificato un certo numero, n , di file da scaricare e abbiamo assegnato un'operazione di download ad ogni PC. Per aumentare la precisione delle misure, lo stesso esperimento è stato ripetuto più volte, utilizzando sempre lo stesso numero n di file, ma cambiando ogni volta il set di file. Per ottenere statistiche comparabili abbiamo selezionato file che hanno circa le stesse dimensioni e popolarità. In particolare, abbiamo considerato file di dimensioni nel range [3,95 GB - 4,71 GB]. Per ogni file il numero iniziale di seed (cioè peer che hanno già l'intero file) era nel range [200 - 800]. Per avere condizioni sperimentali simili, gli esperimenti (con e senza proxy) sono stati alternati, in modo da confrontare risultati ottenuti con condizioni di congestione della rete Internet simili, e che il numero di peer sia simile. Sia le condizioni di Internet, sia il numero di peer interessati ad un file sono variabili non controllabili. Alternando gli esperimenti con e senza

proxy abbiamo cercato di limitare la variabilità di questi parametri tra esperimenti successivi.

4.3.2.2 Risultati sperimentali ottenuti

Illustriamo ora i risultati sperimentali ottenuti, evidenziando sia gli aspetti relativi al risparmio energetico che si ottiene utilizzando EE-BitTorrent, sia gli aspetti relativi alla QoS, ed in particolare alla riduzione dei tempi di download.

4.3.2.2.1 Risparmio energetico

Iniziamo la nostra analisi studiando il risparmio energetico ottenuto con la nostra architettura proxy-based, rispetto a quella legacy.

Numero di PC	$d_L(s)$	$t_1(s)$	$t_2(s)$	$d_p(s)$	$d_p^{\max}(s)$
1	8022,67	0,16	378,41	9288,53	9288,53
2	8927,82	0,32	756,82	5240,03	7596,92
3	6168,82	0,48	1135,23	4130,78	6084,92
4	7719,75	0,42	1246,50	3756,99	7612,89
5	9261,90	0,80	1892,05	6419,13	15012,28
6	10967,55	0,96	2270,45	8725,34	15104,89

Tabella 1 - Valori medi delle componenti temporali.

La Tabella 1 mostra i valori medi degli elementi di ritardo sperimentate da ogni singolo PC e dal proxy BitTorrent con un differente numero di PC (cioè di download di file in parallelo), mentre la Figura 10 mostra il risparmio energetico relativo fornito dalla nostra soluzione proxy-based, rispetto all'architettura legacy. Se il proxy BitTorrent è un PC multi-server che deve rimanere costantemente acceso per altre ragioni (e, quindi, l'energia consumata dal proxy può essere trascurata), il risparmio energetico relativo $S'(n)$ (curva celeste in Figura 10) non dipende dal numero n di PC ed è approssimativamente uguale al 95% per ogni PC. Invece, quando il proxy è una macchina dedicata, quindi il suo consumo energetico va tenuto in considerazione, il risparmio energetico relativo $S''(n)$ (curva rossa in Figura 10) cresce con il numero di PC simultaneamente coinvolti nel download di un file, in modo che i costi del proxy siano suddivisi tra un numero crescente di PC.

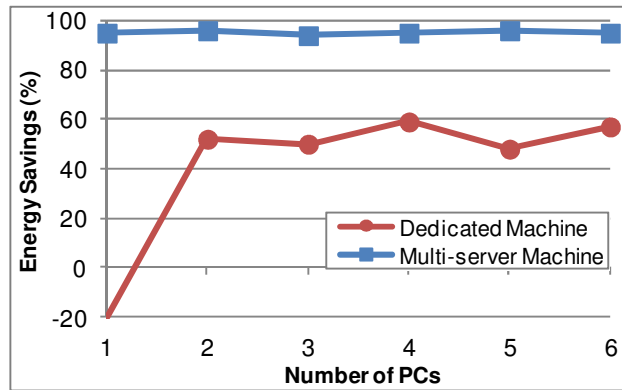


Figura 10 - Risparmio energetico relativo vs numero di PC.

Questi risultati possono essere facilmente spiegati osservando le equazioni (7) e (9). Assumendo che (i) tutti i file abbiano approssimativamente lo stesso tempo di download (cioè, $d_L(i) = d_L, \forall i$), (ii) tutti i client sperimentino approssimativamente lo stesso tempo per inviare la richiesta al proxy BitTorrent e per scaricare il file dal proxy stesso (cioè, $t_1(i) = t_1, t_2(i) = t_2, \forall i$), e (iii) l'interferenza fra client sia trascurabile a causa della larga banda della LAN, l'equazione (7) può essere scritta come

$$S'(n) = 1 - \frac{n \cdot (t_1 + t_2)}{n \cdot d_L} = 1 - \frac{t_1 + t_2}{d_L} = S'(1) \quad (13)$$

Questo significa che, se il proxy è una macchina multi-server, il risparmio energetico relativo ottenuto da ogni singolo PC non dipende dal numero di file scaricati in parallelo. Di conseguenza, il risparmio energetico assoluto cresce (approssimativamente) in modo lineare con il numero di PC. Spieghiamo in dettaglio. Prima, calcoliamo $\Delta E'$ e $\Delta E''$ inserendo le componenti temporali misurate con i nostri esperimenti nelle equazioni (10) e (12), rispettivamente.

Per quanto riguarda il consumo energetico dei PC e del proxy BitTorrent, dobbiamo sottolineare che dipende fortemente dal tipo di computer (per esempio, desktop o computer portatile). Inoltre, per la stessa macchina, varia nel tempo a seconda della modalità operativa. In [54] sono mostrati i valori tipici di consumo energetico di un PC in normale stato di inattività che sono nell'intervallo 70 W – 100 W per macchine desktop e 15 W – 30

W per computer portatili. Pertanto, nel calcolo del risparmio energetico assoluto ottenuto dalla nostra architettura proxy – based abbiamo considerato due differenti valori di consumo energetico, cioè 30 W e 100 W.

I risultati ottenuti sono riportati nella Figura 11. Essa mostra chiaramente che, in entrambi i casi, il risparmio energetico assoluto aumenta quasi linearmente con il numero di PC (ovvero, il numero di file scaricati in parallelo). Per entrambi i valori di consumo presi in considerazione, la differenza tra le due curve corrispondenti è dovuta soltanto al consumo di energia del proxy. Anche in questo caso, è possibile fornire una spiegazione analitica per i comportamenti osservati nella Figura 11.

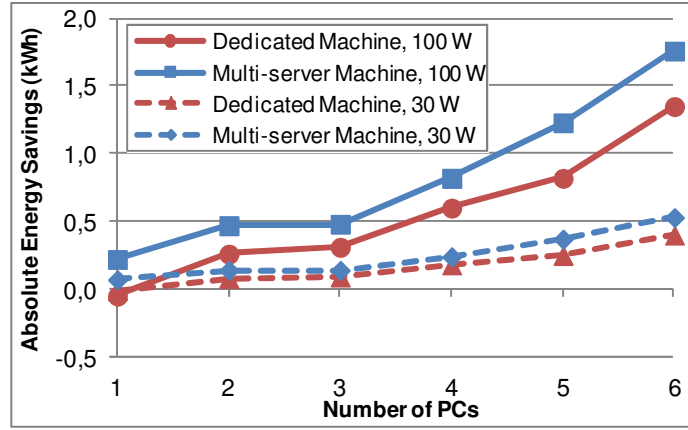


Figura 11 - Risparmio anenergetico assoluto vs numero do PC.

Seguendo la stessa argomentazione utilizzata sopra, è facile dimostrare che $\Delta E'(n)$ aumenta quasi linearmente con il numero di file:

$$\Delta E'(n) \approx n \cdot [d_L - (t_1 + t_2)] \cdot P_{PC} = n \cdot \Delta E'(1) \quad (14)$$

Includendo inoltre l'energia consumata dal proxy nel consumo energetico totale, otteniamo (dall'equazione (12)),

$$\Delta E''(n) \approx \{n \cdot [d_L - (t_1 + t_2)] - d_P^{\max}\} \cdot P_{PC} \quad (15)$$

Assumendo che $d_L \approx d_P^{\max}$ l'equazione (15) può essere approssimata come

$$\Delta E''(n) \approx [(n-1) \cdot d_L - n \cdot (t_1 + t_2)] \cdot P_{PC} \xrightarrow{n} \Delta E'(n) \quad (16)$$

Inoltre, seguendo lo stesso ragionamento abbiamo:

$$S''(n) \approx 1 - \frac{n \cdot (t_1 + t_2) + d_P^{\max}}{n \cdot d_L} \approx 1 - \frac{(t_1 + t_2)}{d_L} - \frac{1}{n} \xrightarrow{n} 1 - \frac{(t_1 + t_2)}{d_L} = S'(n) \quad (17)$$

Le equazioni (16) e (17) mostrano che, per valori elevati di n , il consumo energetico del proxy BitTorrent è trascurabile. Questo andamento non emerge chiaramente dalle Figura 10 e Figura 11 perché il numero di download paralleli considerato (cioè, n) non è abbastanza grande, per ragioni pratiche. Inoltre, anche replicando molte volte ogni singolo esperimento, i risultati ottenuti sono caratterizzati da un'elevata variabilità causata da due ragioni: in primo luogo, le condizioni della rete variano nel tempo. Secondariamente, i file considerati non hanno esattamente la stessa dimensione. Infine, il tempo necessario per scaricare completamente un file è fortemente dipendente dal numero e dalla posizione dei peer che hanno quel file.

4.3.2.2.2 Tempo di download del file

I risultati presentati mostrano chiaramente l'efficacia dell'architettura proxy-based dal punto di vista dell'efficienza energetica. Studiamo ora l'impatto del proxy BitTorrent sulla Quality of Service (QoS) percepita dall'utente, ad esempio, sul tempo di download del file. Mostriamo che l'architettura proxy-based non introduce alcun degrado della QoS. Invece, rispetto all'approccio legacy, riduce significativamente il tempo medio di download dei file. Nell'analisi precedente abbiamo assunto che il tempo per scaricare un file non è significativamente influenzato dalla presenza del proxy. Per analizzare questo aspetto abbiamo effettuato una serie di esperimenti e misurato il tempo necessario per scaricare n file in parallelo, con l'architettura legacy e con quella proxy-based, rispettivamente. I risultati di questa analisi sono riassunti nella Figura 12, dove si vede il tempo medio di download di un singolo file, per un numero crescente di file scaricati in parallelo (ogni colonna in Figura 12 è la media calcolata su tutte le repliche). I risultati mostrati nella Figura 12 indicano chiaramente che il proxy BitTorrent non introduce alcun calo della QoS sperimentata dall'utente. Anzi, in media, il tempo per il download di un file si riduce quando si usa l'architettura proxy-based. Questo può essere spiegato considerando che il

peer in esecuzione sul proxy BitTorrent condivide molte più parti di file rispetto a quelle condivise da ogni singolo peer nell'architettura legacy, e quindi gli viene concessa maggiore larghezza di banda di download. Per quantificare il guadagno medio che si può ottenere quando si usa la nostra architettura proxy-based, abbiamo calcolato il tempo medio per scaricare un file in tutti gli esperimenti condotti. I tempi medi di download erano 6541 s con proxy-based BitTorrent e 8439 s con BitTorrent legacy. Così, il proxy BitTorrent riduce il tempo medio di download del file di circa il 22 %.

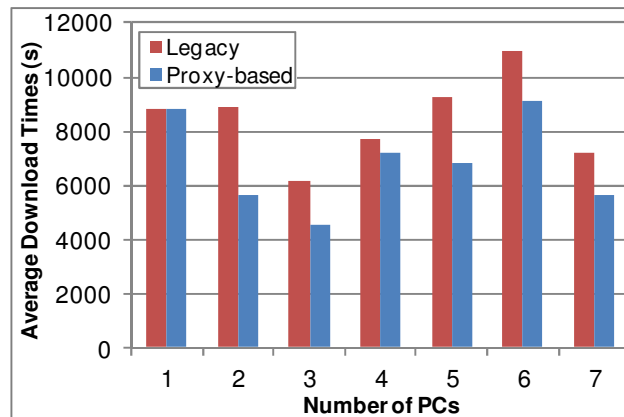


Figura 12 - Tempi medi di download sperimentati da ogni singolo file nell'architettura legacy e proxy-based.

Vogliamo concludere la nostra analisi sul tempo di download di file, sottolineando un interessante aspetto che è strettamente legato al comportamento di BitTorrent. In particolare, analizziamo come la disponibilità sul proxy di un singolo (popolare) file da fornire è in grado di ridurre fortemente il tempo di download di tutti i file che il proxy sta scaricando. Questo effetto è mostrato dai risultati in Figura 13, dove si mostra come il tempo medio sperimentato dal proxy BitTorrent per scaricare lo stesso file è influenzato dalla presenza di un file popolare sul proxy stesso. Abbiamo eseguito due serie di esperimenti nei quali il proxy scaricava, in parallelo, 3 e 4 file diversi, rispettivamente.

Per ogni insieme abbiamo preso in considerazione i due casi in cui un file popolare è e non è disponibile sul proxy. Come emerge chiaramente dai risultati in Figura 13, un singolo file

popolare può ridurre ulteriormente, rispetto al guadagno già ottenuto con il proxy, il tempo di download di tutti gli altri file del 25 % - 30 %.

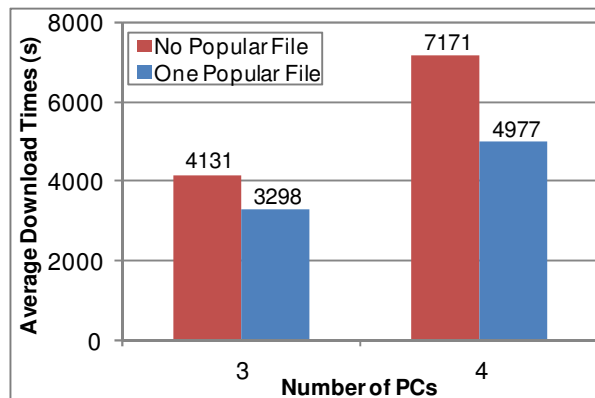


Figura 13 - Impatto, sul tempo di download del proxy, di un file popolare distribuito in parallelo a 3 e a 4 peer.

Oltre all'efficienza energetica, questo fornisce una forte motivazione per sfruttare la nostra architettura proxy-based: un singolo file popolare condiviso sul proxy fornisce un elevato beneficio per tutti. Questo suggerisce di adottare politiche che consistono nella selezione di file popolari disponibili sul client per essere caricati e poi condivisi dal proxy, come un modo per ottimizzare le prestazioni complessive di download del sistema.

4.4 Valutazione sperimentale in uno scenario residenziale

Una valutazione dettagliata delle prestazioni volta a comparare EE-BitTorrent con il protocollo BitTorrent tradizionale, in termini di consumo energetico e di tempo di download di file, è stata condotta nel paragrafo precedente. Si basava sia su analisi che su misure sperimentali ed era focalizzata sugli utenti con accesso a Internet da una rete dipartimentale. In un simile contesto, l'approccio proxy-based si è rivelato più conveniente dell'approccio convenzionale, in quanto riduce significativamente il consumo di energia, senza aumentare il tempo medio necessario per scaricare un file. Tuttavia, la maggior parte degli utenti di BitTorrent accede a Internet da reti di accesso residenziali (es. collegamenti ADSL o UMTS). Pertanto, è estremamente importante indagare le prestazioni di EE-

BitTorrent anche in uno scenario residenziale. A tal fine abbiamo istituito un testbed sperimentale e confrontato le prestazioni ottenute da un utente BitTorrent con l'approccio legacy e con quello proxy-based, rispettivamente.

4.4.1 Analisi dell'efficienza energetica

Per introdurre le metriche per valutare le prestazioni che si prenderanno in considerazione nella nostra analisi sperimentale, consideriamo un insieme di N utenti BitTorrent. Definiamo $D_L(i)$ e $D_P(i)$, $i = 1, 2, \dots, N$, il tempo totale necessario ad un PC utente generico per scaricare il file con l'approccio BitTorrent legacy e proxy-based, rispettivamente. In entrambi i casi, questo tempo è proporzionale all'energia consumata dall'utente per ottenere una copia del file desiderato. Quindi, l'energia risparmiata utilizzando EE-BitTorrent, rispetto ad un approccio legacy, per un generico utente i , può essere espresso come

$$S_{usr}(N) = 1 - \frac{D_P(i)}{D_L(i)} = 1 - \frac{D_r(i) + D_t(i)}{D_L(i)} \approx 1 - \frac{D_t(i)}{D_L(i)} \quad (18)$$

dove $D_r(i)$ è il tempo necessario per inviare la richiesta al proxy, e $D_t(i)$ è il tempo necessario per trasferire il file dal proxy al PC utente. L'ultimo passaggio nell'equazione (18) è dovuto al fatto che $D_r(i) \ll D_t(i)$, per ogni i . Per rendere il confronto equo, è necessario considerare anche l'energia consumata dal proxy, quando si utilizza EE-BitTorrent. Quindi, osservando l'intero sistema (PC utente + proxy), e assumendo che il proxy e il PC abbiano un consumo di potenza simile, il risparmio energetico introdotto da EE-BitTorrent, rispetto all'approccio legacy, può essere espresso come

$$S_{sys}(N) = 1 - \frac{2 \cdot \sum_{i=1}^N D_r(i) + \sum_{i=1}^N D_o(i) + 2 \cdot \sum_{i=1}^N D_t(i)}{D_L(i)} \quad (19)$$

dove $D_o(i)$ indica il tempo impiegato dal proxy per scaricare il file richiesto dall'utente i dalla overlay BitTorrent. Come sopra, dato che $D_r(i) \ll D_t(i), \forall i$, l'equazione (19) può essere riscritta come

$$S_{sys}(N) = 1 - \frac{\sum_{i=1}^N D_o(i) + 2 \cdot \sum_{i=1}^N D_t(i)}{\sum_{i=1}^N D_L(i)} \quad (20)$$

4.4.2 Risultati sperimentali

Passiamo adesso alla descrizione dell'apparato sperimentale utilizzato per la valutazione di EE-BitTorrent nello scenario residenziale ed dei risultati ottenuti distinguendo tra rete di accesso ADSL e rete di accesso UMTS per evidenziarne le caratteristiche peculiari.

4.4.2.1 Descrizione del testbed

Per misurare le componenti temporali introdotte nel paragrafo precedente (cioè, $D_L(i)$, $D_t(i)$ e $D_o(i)$, per $i = 1, 2, \dots, N$), e valutare le equazioni (18) e (20), abbiamo utilizzato un testbed composto da N PC connessi a Internet tramite una rete di accesso residenziale. Nelle prove effettuate utilizzando EE-BitTorrent, il proxy era posizionato all'Università di Pisa ed era connesso a Internet tramite una Ethernet LAN a 100 Mbps. I PC residenziali erano posizionati a molti chilometri di distanza dal proxy (in alcuni casi in città diverse). Abbiamo considerato due diverse tecnologie di accesso per gli utenti BitTorrent, cioè

- accesso ADSL con bit rate nominali pari a 8,0 Mbps (downlink) e 512 Kbps (uplink).
- accesso cellulare UMTS con bit rate nominali pari a 7,2 Mbps (downlink) e 2,0 Mbps (uplink).

Nei nostri esperimenti abbiamo considerato i seguenti file, di diverse dimensioni, che rappresentano le tre categorie tipiche di file frequentemente scaricati dagli utenti BitTorrent

- CD audio (formato mp3, ~100 MB)
- Episodio di una serie TV (formato avi, ~350 MB)
- Distribuzione Ubuntu 10.10 (formato iso, ~4GB)

In tutti gli esperimenti abbiamo considerato lo stesso numero di seed nel torrent (cioè circa 500). Gli esperimenti con BitTorrent legacy e proxy-based erano alternati in modo da sperimentare condizioni operative simili (ad esempio, traffico di rete, numero di peer nel

torrent, ecc.). Infine, per aumentare l'accuratezza dei nostri risultati, abbiamo ripetuto lo stesso esperimento molte volte, in diversi giorni e orari. Per ogni misura mostreremo il valore medio calcolato su tutte le repliche, così come la deviazione standard. I risultati ottenuti con reti residenziali ADSL e UMTS saranno discussi nei sottoparagrafi successivi.

4.4.2.2 Risultati sperimentali ottenuti su rete di accesso ADSL

Iniziamo la nostra analisi prendendo in esame la rete di accesso ADSL e considerando un singolo utente BitTorrent posto nella location A. In Figura 14, come in tutte le figure successive, mostriamo il tempo medio che il PC utente deve rimanere acceso per scaricare una copia del file con l'approccio legacy, cioè $D_L(i)$ (colonna celeste), e quello proxy-based, cioè $D_r(i) + D_i(i) \approx D_i(i)$ (colonna rossa). Per EE-BitTorrent è mostrato anche il tempo medio che il proxy deve rimanere attivo, cioè $D_r(i) + D_o(i) + D_i(i) \approx D_o(i) + D_i(i)$ (colonna verde). La differenza tra i valori relative al proxy e al PC, nella versione proxy-based, rappresenta il tempo impiegato dal proxy per scaricare il file dalla overlay BitTorrent, cioè $D_o(i)$.

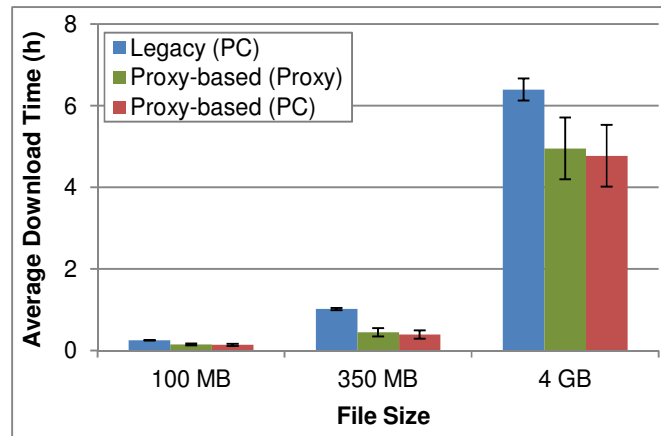


Figura 14 - Tempo medio di download per l'utente nella location A.

I risultati in Figura 14 mostrano che il tempo (medio) che il PC utente deve rimanere acceso per scaricare il file è sempre minore utilizzando l'approccio proxy-based. In particolare, per

il file da 4 GByte il tempo medio di download con EE-BitTorrent è circa il 25 % inferiore (4,8 vs. 6,5 ore). Simili riduzioni, o addirittura più grandi, sono ottenute anche con gli altri due file.

Poi, abbiamo replicato lo stesso insieme di esperimenti con un singolo utente BitTorrent posto in una posizione differente, la location B. I risultati ottenuti sono riassunti in Figura 15.

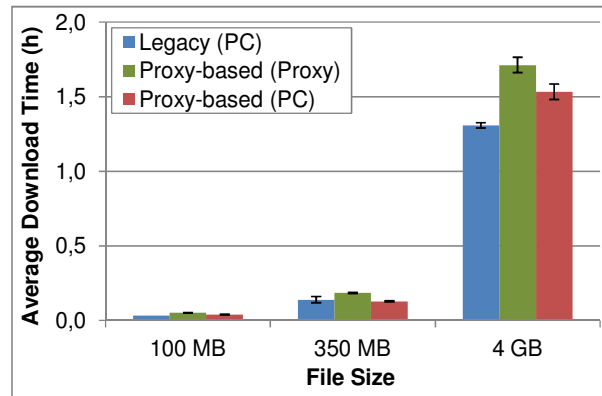


Figura 15 – Tempo medio di download per l'utente nella location B.

La situazione è ora completamente diversa, in quanto l'approccio legacy è sempre più conveniente rispetto a EE-BitTorrent, in termini di tempo medio di download. La completa differenza tra le prestazioni ottenute nelle due location A e B è dovuta al corrispondente throughput in uplink. Sebbene le due location abbiano la stessa banda nominale (in uplink e downlink), il throughput reale disponibile in uplink nella location A è molto minore rispetto a quello disponibile nella location B. A causa della politica Tit-for-Tat utilizzata dal protocollo BitTorrent, i peer con una banda in uplink bassa sono penalizzati e sperimentano un download rate basso che fa aumentare il tempo di download del file (ed il corrispondente consumo energetico). Quando si utilizza EE-BitTorrent il tempo per trasferire il file dal proxy al PC utente dipende dal rate di downlink. Inoltre, dato che il proxy accede a Internet tramite un link a 100 Mbps e può quindi fornire un elevato rate in uplink, esso ottiene solitamente un download rate molto alto dalla overlay BitTorrent. In particolare, per la

location A abbiamo misurato un download rate medio fino a 0,89 Mbps utilizzando l'approccio legacy, e fino a 2,89 Mbps utilizzando EE-BitTorrent. D'altro canto, per la location B il download rate misurato è stato al più 6,66 Mbps per legacy BitTorrent e 6,18 Mbps per EE-BitTorrent.

In termini di energia consumata da un singolo utente, introducendo i precedenti risultati dell'equazione (18), segue che utilizzare EE-BitTorrent è molto benefico per la location A (25,4 % della riduzione rispetto a legacy BitTorrent per il file da 4 GByte), mentre non è così per la location B (16,8% d'incremento rispetto a legacy BitTorrent per il file da 4 GByte). In termini di energia consumata dall'intero sistema (cioè PC utente + proxy), EE-BitTorrent non è mai conveniente. Questo era previsto dato che c'è un singolo PC che utilizza il proxy.

Analizziamo ora il caso (più realistico) in cui ci sono molti utenti BitTorrent che scaricano i loro file in parallelo. In particolare, abbiamo considerato 4 diversi utenti che accedono a Internet da quattro location diverse, chiamate A, B (già introdotte), C e D. Quando ci sono molti utenti che trasferiscono dati dal proxy in parallelo, il tempo totale di trasferimento è uguale al tempo massimo impiegato dal PC più lento, cioè $\sum_{i=1}^N D_i(i) = \max\{D_i(i)\}$. La

Tabella 2 mostra il tempo medio impiegato dall'utente A e dall'utente B (l'utente con il peggior ed il miglior rate di trasferimento, rispettivamente) per scaricare il file da 4 GByte dal proxy, per un numero crescente di PC coinvolti.

Location	1 PC	2 PCs	3 PCs	4 PCs
A	4,77 h (4,01-5,53)	3,72 h (2,96-4,49)	4,27 h (3,72-4,82)	4,55 h (4,40-4,71)
B	1,53 h (1,48-1,58)	1,50 h (1,47-1,54)	1,56 h (1,52-1,61)	1,57 h (1,52-1,62)

Tabella 2 – Tempo medio di download per utenti nelle location A e B con un differente numero di PC che scaricano in parallelo (file da 4 GByte).

Per la stessa location, il tempo di trasferimento non varia significativamente con il numero di PC coinvolti, almeno fino ad un certo numero. Questo accade perché il rate di download

disponibile degli utenti ADSL è molto più basso del rate di upload del proxy. Quindi, l'equazione (20) può essere riscritta come

$$S_{sys}(N) = 1 - \frac{\sum_{i=1}^N D_o(i) + (2 \cdot \max\{D_t(i)\})}{\sum_{i=1}^N D_L(i)} \quad (21)$$

I valori di $S_{usr}(N)$ e $S_{sys}(N)$, calcolati in accordo con le equazioni (18) e (21), rispettivamente, per differenti valori di N , sono mostrati in Tabella 3.

Location		1 PC	2 PCs	3 PCs	4 PCs
A	$S_{usr}^A(N)$	25,4 %	41,8 %	33,2 %	28,8 %
	$S_{sys}(N)$	-52,1 %	-1,2 %	0,8 %	4,8 %
B	$S_{usr}^B(N)$	-16,8 %	-14,5 %	-19,1 %	-19,8 %
	$S_{sys}(N)$	-147,3 %	-1,2 %	0,8 %	4,8 %

Tabella 3 - Risparmio energetico per utente e dell'intero sistema (file da 4 GByte).

Per motivi di spazio mostriamo i risultati soltanto per le location A e B (quelle con i peggiori ed i migliori risultati per EE-BitTorrent) e per il file da 4 GByte. Il risparmio energetico introdotto da EE-BitTorrent per lo specifico utente (cioè, $S_{usr}(N)$ in Tabella 3) non dipende significativamente dal numero di altri PC coinvolti. Esso dipende soltanto dalla specifica location. Le variazioni di $S_{usr}^A(N)$ e $S_{usr}^B(N)$ in Tabella 3, per differenti valori di N , sono dovuti soltanto alle variazioni nel rate ottenute dall'utente nel momento in cui i diversi esperimenti sono stati effettuati (vedi anche Tabella 2). In particolare, il risparmio energetico introdotto da EE-BitTorrent è sempre positivo per l'utente A, mentre è sempre negativo per l'utente B. Al contrario, il risparmio energetico a livello di sistema (cioè $S_{sys}(N)$) è lo stesso per tutti gli utenti (per $N > 1$). Come previsto, esso è fortemente negativo per $N=1$ (perché c'è un singolo PC che utilizza il proxy). Comunque, esso

aumenta rapidamente con il numero di PC coinvolti (cioè utenti che scaricano in parallelo) perché il consumo energetico addizionale dovuto al proxy (rispetto all'approccio legacy) è condiviso tra più utenti.

4.4.2.3 Risultati sperimentali ottenuti su rete di accesso UMTS

In questa sezione abbiamo valutato le prestazioni ottenute da un utente BitTorrent residenziale connesso a Internet tramite un link wireless UMTS. Abbiamo ripetuto lo stesso set di esperimenti con un singolo utente BitTorrent in due diversi periodi di tempo, chiamati P1 e P2. I risultati ottenuti sono riassunti in Figura 16 e Figura 17, rispettivamente. La situazione nei due periodi considerati è estremamente differente. Durante il periodo di tempo P1, la banda disponibile in uplink era estremamente bassa, e questo penalizzava le prestazioni di BitTorrent legacy rispetto a quelle di EE-BitTorrent (il rate medio di download era 90 Kbps e 140 Kbps, rispettivamente). Invece, durante il periodo di tempo P2, la banda disponibile in uplink era molto migliore e l'approccio legacy era molto più efficiente rispetto a EE-BitTorrent.

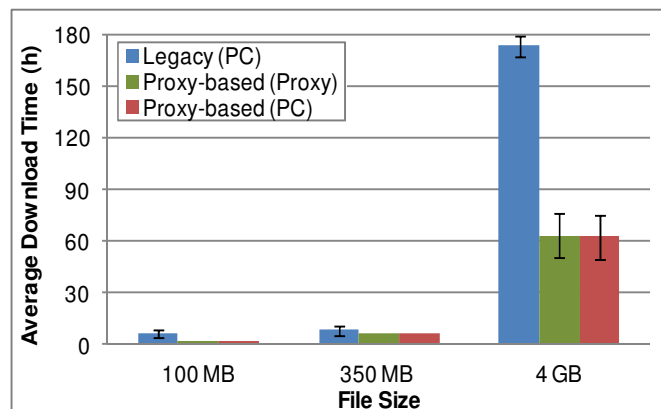


Figura 16 - Tempo medio di download durante il periodo P1.

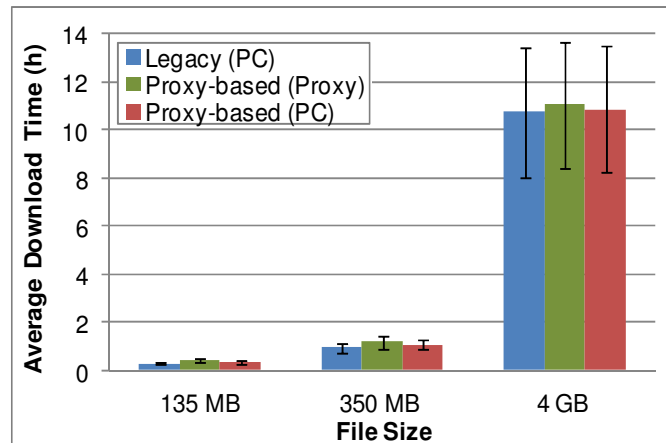


Figura 17 - Tempo medio di download durante il periodo P2.

4.5 Lezione imparata

In questo capitolo abbiamo proposto EE-BitTorrent, una versione proxy-based di BitTorrent orientata all'efficienza energetica. È un meccanismo per risparmiare energia in un contesto in cui gli utenti lasciano il proprio PC costantemente acceso per condividere file tramite BitTorrent. Il problema è legato al fatto che le applicazioni P2P impongono il requisito di una connettività permanente. Per superare questo problema, la nostra architettura EE-BitTorrent offre la possibilità agli utenti BitTorrent di delegare il download dei file ad un proxy. Il nostro obiettivo è il risparmio energetico sul PC dell'utente senza pagarla però con un significativo degrado della QoS, specialmente in termini di tempo di download dei file.

Abbiamo realizzato e valutato la nostra soluzione in due diversi testbed realistici: uno scenario dipartimentale / aziendale, ed uno scenario residenziale. Abbiamo misurato il tempo di download dei file utilizzando sia l'approccio legacy che quello proxy-based.

I risultati sperimentali ottenuti mostrano che, nello scenario dipartimentale, l'architettura proxy-based consente di ottenere un risparmio energetico fino al 95 % rispetto all'architettura legacy. Questo mostra l'efficacia, dal punto di vista energetico, della nostra soluzione in questo scenario. Inoltre, l'utilizzo del proxy non introduce alcun degrado della QoS, anzi, il tempo medio di download dei file si riduce di circa il 22 % utilizzando la

versione proxy-based di BitTorrent dato che il numero di file condivisi dal proxy è maggiore di quello dei file che sono solitamente condivisi da un singolo peer.

Nello scenario residenziale invece, i risultati ottenuti mostrano che il download rate disponibile per gli utenti BitTorrent è fortemente dipendente dalla specifica location. Inoltre, per la stessa location, il download rate disponibile può variare significativamente nel tempo. Abbiamo anche visto che, a seconda del throughput disponibile in downlink, utilizzare EE-BitTorrent non comporta necessariamente un tempo di download più basso ed un minor consumo energetico rispetto al protocollo BitTorrent convenzionale. In scenari residenziali la situazione è quindi differente rispetto a quella dello scenario dipartimentale dove, in accordo coi risultati sperimentali mostrati, EE-BitTorrent tipicamente supera le prestazioni del protocollo legacy BitTorrent, grazie all'elevata banda generalmente disponibile in questo scenario. Invece, in uno scenario residenziale, l'approccio più conveniente non può essere deciso a priori, in quanto dipende, in ultima analisi, dalle condizioni operative della rete di accesso residenziale. Motivati da queste osservazioni, nel prossimo capitolo definiremo un Adaptive BitTorrent (AdaBT), un algoritmo adattivo che seleziona l'opzione più conveniente (tra BitTorrent tradizionale e EE-BitTorrent), in funzione delle condizioni operative attuali.

Capitolo 5

Adaptive BitTorrent (AdaBT)

Nel capitolo precedente abbiamo misurato sperimentalmente le prestazioni, in termini di efficienza energetica e di tempo di download dei file, di EE-BitTorrent in due scenari con caratteristiche molto diverse: lo scenario dipartimentale / aziendale, in cui i PC sono collegati attraverso una rete LAN ad alta velocità (cioè, 100 Mbps Ethernet), la stessa su cui si trova il proxy, e lo scenario residenziale, con rete di accesso ADSL e UMTS, da posizioni diverse e con diversi Internet Service Provider (ISP), e la cui larghezza di banda disponibile è molto inferiore a quella di una LAN dipartimentale.

I risultati ottenuti dimostrano che nel primo contesto, la soluzione proxy-based di BitTorrent consente di risparmiare fino al 95 % di energia a causa della riduzione dei tempi di download dei file e può quindi essere adottata in sostituzione della versione tradizionale. Nel secondo contesto invece, essendo il tempo richiesto per il download di un file, e il consumo energetico corrispondente, fortemente influenzato dalla velocità disponibile in uplink (a causa della strategia Tit-for-Tat) con il protocollo legacy BitTorrent, non sempre e ovunque EE-BitTorrent offre prestazioni migliori di BitTorrent legacy. Questo accade perché, su reti di accesso residenziale, la banda disponibile varia in funzione della posizione geografica e del tempo. Quindi, una semplice architettura proxy-based potrebbe non essere ottimale per tutti gli utenti residenziali.

Motivati da questi risultati sperimentali, in questo capitolo definiamo Adaptive BitTorrent (AdaBT), un algoritmo adattivo che è in grado di selezionare dinamicamente l'opzione più conveniente (tra BitTorrent convenzionale e EE-BitTorrent) a seconda delle condizioni operative effettive disponibili nel tempo di download. In quanto tale, il protocollo AdaBT è un'estensione di EE-BitTorrent e aggiunge flessibilità.

5.1 Algoritmo AdaBT

AdaBT ha un approccio molto semplice. Alla ricezione di una richiesta da parte dell'utente, misura la velocità di download che può essere raggiunta dalla rete overlay di BitTorrent e dal proxy BitTorrent. Poi, sulla base di queste misurazioni, seleziona l'opzione più conveniente. Quando l'opzione selezionata è EE-BitTorrent, se il file richiesto è già disponibile sul proxy (che è un caso molto frequente), il trasferimento dati al PC dell'utente può iniziare immediatamente. In caso contrario, l'utente BitTorrent è invitato a spegnere il PC (per il risparmio energetico) e ricollegarsi in seguito per scaricare il file dal proxy.

Algoritmo 1: Algoritmo AdaBT

```

1  AdaBT_start_time = time();

2a  Start Legacy BitTorrent; wait for the first chunk of data;

3a  Legacy_start_time=time();
4a  Wait for Q data;
5a  Legacy_end_time = time();
6a   $T_L = \text{Legacy\_end\_time} - \text{Legacy\_start\_time}$ 
7a   $R_L = Q/T_L$  // estimated rate with legacy protocol

2b  request S data from Proxy;
3b  Proxy_start = time();
4b  Wait for Q data;
5b  Proxy_end_time = time();
6b   $T_P = \text{Proxy\_end\_time} - \text{Proxy\_start\_time}$ 
7b   $R_P = Q/T_P$  // estimated rate from proxy

8  AdaBT_end_time=time();
9   $T_{OH} = \text{AdaBT\_end\_time} - \text{AdaBT\_start\_time}$ ;
10 if  $(L/R_P + T_{OH}) < \alpha \cdot ((L-Q)/R_L)$  then use EE-BitTorrent

11 else use legacy BitTorrent;
```

L'Algoritmo 1 mostra le azioni dettagliate eseguite da AdaBT. Alla ricezione di una richiesta da parte dell'utente, AdaBT inizia stimando la velocità di download attualmente disponibile dalla rete overlay BitTorrent (linee 2a-7a) e dal proxy BitTorrent configurato (linee 2b-7b). Queste due azioni sono svolte in parallelo. Quando si usa BitTorrent legacy,

l'algoritmo attende che il primo blocco di dati venga ricevuto prima di iniziare a stimare il rate di download disponibile. In entrambi i casi, il rate viene stimato scaricando una quantità di byte Q predefinita e misurando il tempo corrispondente. Nel prendere la decisione circa l'opzione più conveniente (righe 10-11), AdaBT prende in considerazione l'overhead introdotto dalla fase preliminare di stima della banda. EE-BitTorrent viene selezionata solo quando il throughput stimato per scaricare dal proxy è significativamente più grande del throughput stimato per scaricare dalla rete overlay. Questo è il motivo dell'utilizzo del parametro α nella riga 10. Nei nostri esperimenti (che saranno discussi più avanti) abbiamo usato $\alpha = 0,9$.

Una questione importante nella progettazione dell'algoritmo AdaBT è legata all'impostazione di Q , cioè il numero di byte da cui la stima del throughput dovrebbe essere derivata. Naturalmente, un valore basso per Q potrebbe compromettere la precisione della stima. Dall'altro lato, questo parametro ha un impatto diretto sull'overhead introdotto da AdaBT. A questo proposito, l'overhead dovrebbe essere molto piccolo, rispetto al tempo totale impiegato per scaricare il file desiderato. Tuttavia, questo parametro non può essere conosciuto in anticipo in quanto dipende dalle condizioni di funzionamento, come evidenziato anche nella sezione precedente. Naturalmente, diversi criteri possono essere utilizzati per impostare il valore appropriato di Q . Nella nostra implementazione abbiamo deciso di utilizzare EE-BitTorrent solo quando la dimensione del file è superiore o uguale ad una soglia predefinita. Inoltre, quando si utilizza AdaBT, il valore di Q viene così ricavato come $Q = \min\left\{\frac{L}{10}, Q_{\max}\right\}$, dove L è la dimensione del file da scaricare e $Q_{\max}$ è un'altra soglia predefinita ed è utilizzata per limitare la durata della fase di stima della banda quando il file è molto grande.

5.2 Valutazione sperimentale di AdaBT

In questo paragrafo valutiamo le performance di AdaBT e le confrontiamo con entrambi legacy BitTorrent e EE-BitTorrent. A tal fine, abbiamo usato lo stesso testbed di prova introdotto nel capitolo precedente. In particolare, abbiamo limitato la nostra analisi al caso di un utente BitTorrent singolo con accesso ad Internet attraverso un collegamento ADSL,

sia dalla posizione A che dalla posizione B. Abbiamo preso in considerazione le dimensioni degli stessi file utilizzati nel capitolo precedente (cioè, 100 MByte, 350 MByte, e 4 GByte), e lo stesso numero di seed (circa 500). Inoltre, per AdaBT, abbiamo impostato i seguenti valori dei parametri: $\alpha = 0,9$ e $Q_{\max} = 10$ MByte.

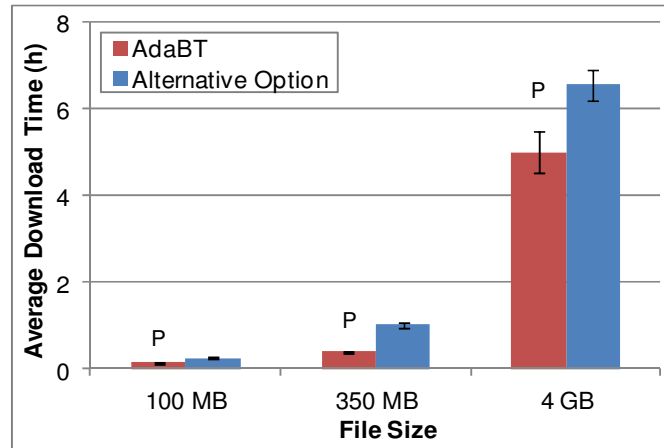


Figura 18 - Tempo medio di download fornito da AdaBT nella location A (file da 4 GByte).

La Figura 18 mostra il tempo medio di download fornito da AdaBT all'utente nella location A. Questo tempo include anche l'overhead introdotto da AdaBT per stimare il throughput disponibile e selezionare l'opzione più efficiente. Per ogni dimensione del file considerata, la Figura 18 mostra anche l'opzione selezionata da AdaBT per scaricare il file, cioè l'approccio legacy (L) oppure quello proxy-based (P). Infine, esso mostra anche il tempo medio di download fornito dall'opzione alternativa (cioè l'opzione non selezionata da AdaBT). Per la location A, AdaBT sceglie sempre l'opzione proxy-based (questo è in accordo con i risultati mostrati nel capitolo precedente per la stessa location). Nonostante l'overhead introdotto, AdaBT è in grado di fornire una significativa riduzione del tempo di download rispetto all'approccio legacy. Abbiamo ripetuto la stessa serie di esperimenti per l'utente in posizione B. In questa posizione i rate disponibili dal proxy e dalla rete overlay BitTorrent sono simili. Quindi, per lo stesso file, AdaBT sceglie in genere una diversa opzione BitTorrent in diverse repliche dell'esperimento, a seconda delle condizioni operative al momento della replica.

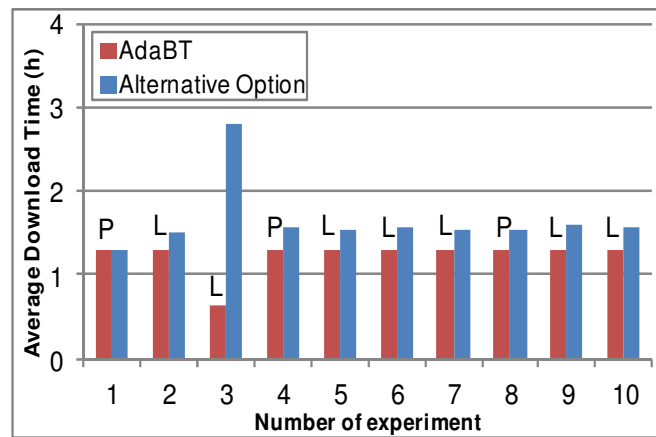


Figura 19 - Tempo medio di download fornito da AdaBT nella location B (file da 4 GByte).

La Figura 19 mostra il tempo di download sperimentato da AdaBT in repliche differenti dell'esperimento con il file da 4 GByte (i risultati per gli altri file sono simili). L'opzione BitTorrent selezionata varia con le repliche degli esperimenti, tuttavia, AdaBT seleziona sempre la versione più efficiente (date le condizioni operative attuali), come evidenziato dal confronto con l'opzione alternativa nella Figura 19. Nonostante l'overhead introdotto, AdaBT è in grado di fornire una significativa riduzione del tempo di download rispetto all'approccio legacy.

Capitolo 6

Conclusioni

In questa tesi abbiamo proposto un proxy BitTorrent per l'efficienza energetica nell'ambito della condivisione di file P2P. È un meccanismo per risparmiare energia in un ambiente in cui molti utenti lasciano il proprio PC costantemente acceso per condividere file tramite BitTorrent. Il problema è dovuto ai requisiti di connettività permanente necessari per il funzionamento delle applicazioni P2P che rendono impossibile l'uso delle tradizionali tecniche di power management. Per superare questo problema, abbiamo proposto un'architettura BitTorrent proxy-based in cui il download dei file viene delegato ad un proxy. Il nostro obiettivo era il risparmio energetico sul PC dell'utente senza però l'introduzione di un significativo degrado della QoS, in particolare senza aumentare il tempo di download dei file. Abbiamo valutato la nostra soluzione in un testbed reale misurando il tempo di download dei file sia con l'architettura legacy sia con quella proxy-based. Dapprima abbiamo condotto le misure in uno scenario dipartimentale / aziendale con i peer BitTorrent posti sulla stessa LAN sulla quale si trovava il proxy e i nostri risultati sperimentali hanno evidenziato che la nostra architettura proxy-based Energy Efficient BitTorrent (EE-BitTorrent) consente di risparmiare fino al 95% dell'energia consumata da ogni PC rispetto all'impiego di BitTorrent legacy. Questo mostra l'efficacia del nostro approccio dal punto di vista dell'efficienza energetica. I nostri risultati mostrano inoltre che utilizzando il proxy BitTorrent non c'è alcun degrado della QoS. Anzi, il tempo di download dei file si riduce di circa il 22 %, dato che il numero di file condivisi con la rete overlay BitTorrent dal proxy è maggiore rispetto al numero di file condiviso da un singolo peer. Infine, abbiamo anche osservato che la presenza sul proxy di un file molto popolare in upload, mentre il proxy sta scaricando i file per i suoi client, introduce un'ulteriore riduzione del tempo medio di download del file del 25 % - 30 %, mostrando ulteriori benefici nell'utilizzo della nostra architettura proxy-based.

In seguito abbiamo valutato la stessa soluzione in ambito residenziale con rete di accesso sia ADSL sia UMTS e abbiamo mostrato che in questo scenario, con singoli PC che si

connettono alla overlay dalle abitazioni degli utenti e si appoggiano ad un proxy posizionato su una rete veloce, ad esempio quella dipartimentale, le prestazioni ottenute sono fortemente influenzate dalla specifica posizione e, per la stessa posizione, possono variare molto nel tempo. In particolare, quando la banda disponibile è bassa, EE-BitTorrent è migliore della versione tradizionale di BitTorrent in termini di tempi medi di download dei file e di consumo energetico del PC dell'utente. Motivati da questi risultati, abbiamo successivamente definito AdaBT, un algoritmo adattivo che è in grado di selezionare, per uno specifico utente, l'opzione BitTorrent più efficiente (tra quella convenzionale e quella proxy-based), in funzione delle condizioni operative attuali sperimentate dall'utente in uno specifico istante. Abbiamo valutato AdaBT in ambito sperimentale e i risultati ottenuti mostrano che AdaBT seleziona sempre l'opzione BitTorrent migliore, cioè quella che apporta il minor dispendio energetico e che quindi introduce una significativa riduzione del tempo di download dei file.

In letteratura, nello specifica ambito delle applicazioni P2P ed in particolare BitTorrent, non sono presenti molti lavori che trattano dell'efficienza energetica, in quanto essendo BitTorrent un protocollo per la condivisione di file, la maggior parte degli sforzi sono volti a migliorarne le prestazioni in termini di riduzione dei tempi di download del file, senza però cercare un compromesso o una miglioria anche riguardo al consumo energetico associato. I lavori che trattano l'argomento sono stati citati nel capitolo 2 sullo stato dell'arte delle soluzioni per il risparmio energetico in Internet. In particolare, rispetto a Green BitTorrent proposto in [64] che sfrutta il meccanismo di risveglio WoL e che necessita quindi di una scheda di rete che sia fornita di tale tecnologia, la nostra soluzione non necessita di hardware specifico ed eventualmente aggiuntivo. Le soluzioni proxy-based proposte in [50, 51] si riferiscono ad una versione semplificata di BitTorrent per ottenere efficienza energetica sui dispositivi mobili, mentre la nostra proposta è riferita ai PC fissi. Infine, le altre soluzioni proxying citate sono di tipo generale e non customizzate per la specifica applicazione P2P BitTorrent.

Bibliografia

- [1] K. Kawamoto, J. Koomey, B. Nordman, R. Brown, M. Piette, M. Ting, A. Meier, "Electricity Used by Office Equipment and Network Equipment in the US", *Energy--The International Journal* (also LBNL-45917), Vol. 27, n. 3, 2002, pp. 255-269.
- [2] C. Webber, J. Roberson, R. Brown, C. Payne, B. Nordman, J. Koomey, "Field Surveys of Office Equipment Operating Patterns", Report LBNL-46930, Energy Analysis Department, Lawrence Berkeley National Laboratory, Settembre 2001, 33 pgs.
- [3] P. Bertoldi, B. Atanasiu, "Electricity Consumption and Efficiency Trends in European Union - Status Report 2009", JRC Technical and Scientific Reports, 2009.
- [4] J. Baliga, K. Hinton, R. Tucker, "Energy Consumption of the Internet", Atti del convegno "COIN-ACOFI" 32nd Australian Conference on Optical Fibre Technology, Melbourne Giugno 2007, pp. 1-3.
- [5] "Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431", U.S. Environmental Protection Agency ENERGY STAR Program, Agosto 2007, 133 pgs.
- [6] "Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO₂ Emissions", Press Release, Stamford, Aprile 2007.
- [7] J. Mankoff, R. Kravets, E. Blevis, "Some Computer Science Issues in Creating a Sustainable World", *IEEE Computer*, Vol. 41, n.8, Agosto 2008, pp. 102-105.
- [8] R. Bolla, R. Bruschi, K. Christensen, F. Cucchietti, F. Davoli, S. Singh, "The Potential Impact of Green Technologies in Next Generation Wireline Networks-Is There Room for Energy Savings Optimization?", *IEEE Communication Magazine (COMMAG)*, Special Topic in Green Communications, Vol. 49, n. 8, Agosto 2011, pp. 80-86.
- [9] C. Gunaratne, K. Christensen, S. Suen, B. Nordman, "Reducing the Energy Consumption of Ethernet with an Adaptive Link Rate (ALR)", *IEEE Transactions on Computers*, Vol. 57, n. 4, Aprile 2008, pp. 448-461.
- [10] S. Karayi, "The PC energy report", IE National Energy Foundation (NEF), Londra, 2007.
- [11] H. Schulze, K. Mochalski, "The Impact of Peer-To-Peer file sharing, voice over IP, Skype, Joost, Instant Messaging, One-Click Hosting and Media Streaming such as YouTube on the Internet", IPOQUE-Internet Study 2007, Leipzig (Germania), Settembre 2007.
- [12] C. Gunaratne, K. Christensen, B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed", *International Journal of Network Management*, Vol. 15, n. 5, Ottobre 2005, pp. 297-310.

- [13] A. Odlyzko, "Data Networks are Lightly Utilized, and will Stay that Way", *Review of Network Economics*, Vol.2, n. 3, Settembre 2003, pp. 210-237.
- [14] K. Christensen, P. Gunaratne, B. Nordman, A. George, "The Next Frontier for Communications Networks: Power Management", *Computer Communications*, Vol. 27, n. 18, Dicembre 2004, pp. 1758-1770.
- [15] K. Christensen, B. Nordman, R. Brown, "Power Management in Networked Devices," (Communications column) *IEEE Computer*, Vol. 37, n. 8, Agosto 2004, pp. 91-93.
- [16] C. Gunaratne, K. Christensen, "Ethernet Adaptive Link Rate: System Design and Performance Evaluation", Atti del convegno "LCN 2006" IEEE Conference on Local Computer Networks, Tampa - Florida (USA), Novembre 2006, pp. 28-35.
- [17] IEEE 802.3az Energy Efficient Ethernet Task Force.
- [18] C. Gunaratne, K. Christensen, S. Suen, "Ethernet Adaptive Link Rate (ALR): Analysis of a Buffer Threshold Policy", Atti del convegno "IEEE Globecom", San Francisco - California (USA), Novembre 2006.
- [19] K. Christensen, F. Blanquicet, "An initial performance evaluation of Rapid PHY Selection (RPS) for Energy Efficient Ethernet", Atti del convegno "LCN 2007" IEEE Conference on Local Computer Networks, Clontarf Castle - Dublin (Ireland), Ottobre 2007, pp. 223-225.
- [20] R. Hays, "Active/Idle Toggling with Low-Power Idle", IEEE 802.3az Task Force Interim Meeting, Gennaio 2008.
- [21] M. Gupta, S. Grover, S. Singh, "A Feasibility Study for Power Management in LAN Switches", Atti del convegno "IEEE ICNP", Berlino (Germania), Ottobre 2004.
- [22] S. Singh, M. Gupta, "Using low-power modes for energy conservation in Ethernet LANs", Atti del convegno "IEEE INFOCOM 2007" 26th IEEE International Conference on Computer Communications IEEE, Anchorage (Alaska) USA, Maggio 2007.
- [23] Y. Fukuda, T. Ikenaga, H. Tamura, M. Uchida, K. Kawahara, Y. Oie, "Dynamic Link Control with Changing Traffic for Power Saving", Atti del convegno "LCN" IEEE Local Computer Networks Conference, Zurich (Switzerland), Ottobre 2009.
- [24] Y. Fukuda, T. Ikenaga, H. Tamura, M. Uchida, K. Kawahara, Y. Oie, "Performance Evaluation of Power Saving Scheme with Dynamic Transmission Capacity Control", 2nd International Workshop on Green Communications (GreenComm) in conjunction with the IEEE GLOBECOM, Honolulu - Hawaii (USA), Novembre 2009.
- [25] A. Chen, T. Wang, R. Katz, "Energy Efficient Ethernet Encodings", Atti del convegno "LCN" IEEE Local Computer Networks Conference, Montreal - Quebec (Canada), Ottobre 2008, pp. 122-129.
- [26] M. Gupta, S. Singh, "Greening of the Internet", Atti del convegno "ACM SIGCOMM", Karlsruhe (Germania), Agosto 2003, pp. 19-26.

- [27] M. Gupta, S. Singh, "Dynamic Ethernet Link Shutdown for Power Conservation on Ethernet Links", Atti del convegno "IEEE ICC" IEEE International Conference on Communications 2007, Glasgow (Scozia), Giugno 2007.
- [28] F. Blanquicet, "PAUSE Power Cycle: A New Backwards Compatible Method to Reduce Energy Use of Ethernet Switches", White Paper, Version 1.0, *Ethernet Alliance*, Aprile 2008.
- [29] R. Bolla, R. Bruschi, A. Ranieri, "Energy-Aware Equipment for Next-Generation Networks", Atti del convegno "ACM CFI", Seoul (Korea), Giugno 2009.
- [30] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsang, S. Wright, "Power Awareness in Network Design and Routing", Atti del convegno "IEEE INFOCOM", Phoenix - AZ (USA), Aprile 2008.
- [31] L. Chiaraviglio, M. Mellia, F. Neri, "Energy-Aware Networks: Reducing Power Consumption By Switching Off Network Elements", FEDERICA Phosphorus Tutorial and Workshop "TNC2008", Brugges (Belgio), Maggio 2008.
- [32] E. Gelenbe, S. Silvestri, "Optimisation of Power Consumption in Wired Packet Networks, Lecture Notes of the Institute for Computer Sciences", **QSHINE**, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Social Informatics and Telecommunications Engineering, Vol. 22, Springer, 2009, pp. 717-729.
- [33] M. Baldi, Y. Ofek, "Time for a 'Greener Internet'", 1st International Workshop on Green Communications (GreenComm) in conjunction with the IEEE International Conference on Communications (IEEE ICC 2009), Dresden (Germania), Giugno 2009.
- [34] J. Restrepo, C. Gruber, C. Mas Machuca, "Energy Profile Aware Routing", 1st International Workshop on Green Communications IEEE International Conference on Communications (IEEE ICC 2009), Dresden (Germania), Giugno 2009.
- [35] L.A. Barroso, U. Holzle, "The Case for Energy-Proportional Computing", *IEEE Computer*, Vol. 40, n. 12, 2007, pp. 33-37.
- [36] J. Chase, R. Doyle, "Balance of Power: Energy Management in Server Clusters", Atti del convegno 8th Workshop on Hot Topics in Operating Systems, Elmau/Oberbayern (Germania), Maggio 2001.
- [37] M. Jimeno, K. Christensen, B. Nordman, "A Network Connection Proxy to Enable Hosts to Sleep and Save Energy", Atti del convegno "IEEE ICC 2008" IEEE International Performance Computing and Communications Conference, Beijing (Cina), Dicembre 2008, pp. 101-110.
- [38] B. Nordman, K. Christensen, "Improving the Energy Efficiency of Ethernet-Connected Devices: A Proposal for Proxying", White Paper, Version 1.0, *Ethernet Alliance*, Ottobre 2007.
- [39] G. Newsham, D. Tiller, "A Case Study of the Energy Consumption of Desktop Computers", Atti del convegno IEEE Industry Applications Society Annual Conference, Houston - Texas (USA), 1992, pp. 1218-1221.

- [40] K. Christensen, "The Next Frontier for Communications Networks: Power Management," Atti del convegno "SPIE" Performance and Control of Next-Generation Communications Networks, Vol. 5244, Settembre 2003, pp. 1-4.
- [41] Magic Packet Technology, White Paper, Novembre 1995.
- [42] http://en.wikipedia.org/wiki/Wake-on-LAN#cite_note-10
- [43] G. Anastasi, M. Conti, E. Gregori, A. Passarella, "Performance Comparison of Power Saving Strategies for Mobile Web Access", *Performance Evaluation*, Vol. 53, Issue 3-4, Agosto 2003, pp. 273-294.
- [44] M. Allman, K. Christensen, B. Nordman, V. Paxson, "Enabling an Energy-Efficient Future Internet Through Selectively Connected End Systems", "HotNets-VI" 6th Workshop on Hot Topics in Networks, Novembre 2007.
- [45] E. Pitoura, G. Samaras, "Data management for Mobile Computing", Kluwer Academic Publishers, 1998.
- [46] UPnP Low Power Architecture V1.0, UPnP Forum, Agosto 2007.
- [47] UPnP Device Architecture, UPnP Forum.
- [48] M. Jimeno, "The SIP Catcher: a service to enable IP phones to sleep", <http://www.youtube.com/watch?v=KdAm4olcVoo>.
- [49] M. Jimeno, K. Christensen, "A Prototype Power Management Proxy for Gnutella Peer-to-Peer File Sharing", Atti del convegno "IEEE LCN 2007" IEEE Conference on Local Computer Networks, Clontarf Castle - Dublin (Irlanda), Ottobre 2007, pp. 210-212.
- [50] I. Kelenyi, A. Ludanyi, J. Nurminen, "BitTorrent on Mobile Phones-Energy Efficiency of a Distributed Proxy Solution", Atti del convegno International Green Computing Conference (IGCC 2010), Chicago, USA, Agosto 2010.
- [51] I. Kelenyi, A. Ludanyi, J. Nurminen, I. Pusstinen, "Energy-efficient Mobile BitTorrent with Broadband Router Hosted Proxies", Atti del convegno "WMNC 2010" IFIP Wireless and Mobile Networking Conference, Budapest (Ungheria), Ottobre 2010.
- [52] K. Christensen, F. Gullede, "Enabling power management for network-attached computers", *International Journal of Network Management*, Vol. 8, n 2, Marzo 1998, pp. 120-130.
- [53] S. Nedeveschi, J. Chandrashekar, B. Nordman, S. Ratnasamy, N. Taft, "Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems", Atti del convegno "NSDI 2009" USENIX Symposium on Networked System Design and Implementation, Boston (USA), Aprile 2009.
- [54] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage", Atti del convegno "NSDI 2009" USENIX Symposium on Networked System Design and Implementation, Boston (USA), Aprile 2009.

- [55] <http://www.gumstix.com/>
- [56] S. Cheshire, "Method and apparatus for implementing a sleep proxy for services on a network", United States Patent 7, 330, 986, Febbraio 2008.
- [57] Y. Agarwal, S. Savage, R. Gupta, "SleepServer: Energy Savings for Enterprise PCs by Allowing them to Sleep", Atti del convegno USENIX Annual Technical Conference, Boston (USA), Giugno 2010.
- [58] Standard ECMA-393 ProxZzzy1 for Sleeping Hosts, 1st edition, Febbraio 2010.
- [59] A. Dalton, C. Ellis, "Sensing User Intention and Context for Energy Management", Atti del convegno "HotOS IX" The 9th Workshop on Hot Topics in Operating Systems, pp. 151-156, Lihue - Hawaii (USA), Maggio 2003.
- [60] F. Albinali, C. Gniady, "CPM: Context-Aware Power Management in WLANs", Atti del convegno "NCAI 2006" 21st National Conference on Artificial Intelligence, Boston - Massachusetts (USA), 2006.
- [61] C. Harris, V. Cahill, "Power Management for Stationary Machines in a Pervasive Computing Environment", Atti del convegno "HICSS 2005" 38th Hawaii International Conference on System Sciences, Hilton Waikoloa Village - Hawaii (USA), 2005.
- [62] <http://www.youtube.com/watch?v=hG5lH8pKqeg>
- [63] L. Irish, K. Christensen, "A 'Green TCP/IP' to Reduce Electricity Consumed by Computers", Atti del convegno "IEEE Southeastcon", pp. 302-305, Aprile 1998.
- [64] J. Blackburn, K. Christensen, "A Simulation Study of a New Green BitTorrent", Atti del convegno "GreenComm 2009" 1st International Workshop on Green Communications, Dresden (Germania), Giugno 2009.
- [65] G. Anastasi, M. Conti, I. Giannetti, A. Passarella, "Design and Evaluation of a BitTorrent Proxy for Energy Saving", Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2009), Sousse, Tunisia, July 5-8, 2009.
- [66] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, "A survey and comparison of Peer to Peer overlay network schemes", *IEEE Communication Surveys and Tutorials*, Vol. 7, n. 2, 2005, pp. 72-93.
- [67] J. Kurose, K. Ross, **Peer-to-Peer Applications, Computer Networking. A Top-Down Approach**, IV Edition, Addison Wesley, 2007.
- [68] D. Towsley, "The Internet is Flat: A brief history of networking over the next ten years", Atti del convegno "ACM PODC 2008", Toronto (Canada), Agosto 2008.
- [69] G. Anastasi, I. Giannetti, A. Passarella, "A BitTorrent Proxy for Green Internet File Sharing: Design and Experimental Evaluation", *Computer Communications*, Vol. 33, n. 7, Maggio 2010, pp. 794-802.

Appendice A

Specifiche del protocollo BitTorrent

CONVENZIONI

- Peer / client: un peer è ogni BitTorrent client partecipante ad un download; il client è anch'esso un peer, ma quello sulla macchina locale.
- Parte / blocco: parte si riferisce alla porzione di dati scaricata menzionata nel file di metainformazioni e verificata con l'hash SHA1; blocco si riferisce alla porzione di dati che un client richiede ad un peer; i blocchi sono parte delle parti.

BENCODING

Codifica che permette di specificare ed organizzare i dati in un formato chiaro.

Supporta 4 tipi: stringhe, interi, liste e dizionari.

- Stringhe: *<lunghezza della stringa in base 10>:<stringa>*, senza delimitatori; es. *spam* è codificata da *4:spam*.
- Interi: *i<intero in base 10>e*; es. *-3* è codificato *i-3e*, *0* è codificato *i0e*, *7* è codificato *i7e*.
- Liste: *l<valore di tipo bencode>e*; es. *['spam', 'mail', 5]* è codificata *l4:spam4:maili5ee*.
- Dizionari: sono coppie chiave-valore; *d<chiave, cioè stringa codificata bencode><valore di tipo bencode>e*; es. *{'mucca' => 'muu', 'spam' => ['a', 'b']}* è codificato *d5:mucca3:muu4:spaml1:a1:bee*.

Implementazioni: in Python esiste un modulo aggiuntivo che decodifica correttamente dati codificati Bencode, dato che il file *bencode.py* nella directory di BitTorrent non gestisce dati bencode annidati ed è 5-6 volte più veloce dell'implementazione in Perl.

STRUTTURA DEL FILE DI METAINFORMAZIONI

Il file di metainformazioni è il *file.torrent*.

Tutti i dati contenuti nel file di metainformazioni sono codificati con codifica Bencode. In particolare, il contenuto è un dizionario Bencode contenenti le seguenti chiavi:

- *info*: dizionario che descrive il o i file del torrent; ci sono due possibili forme: una per il caso di file singolo, senza la struttura a directory, ed una per il caso di file multipli;
- *announce*: URL del tracker (stringa);
- *announce-list*: (opzionale) estensione delle specifiche ufficiali, compatibile anche con versioni precedenti. Questa chiave è utilizzata per implementare una lista di tracker di backup (lista);
- *creation date*: (opzionale) momento di creazione del torrent in secondi (intero);
- *comment*: (opzionale) commento dell'autore del torrent in forma testuale libera (stringa);
- *created by*: (opzionale) nome e versione del programma usato per creare il file.torrent (stringa).

Il dizionario rappresentato dalla chiave *info* contiene i seguenti campi, comuni ai modi singolo file e multipli file:

- *piece length*: numero di byte in ogni parte. Tale valore specifica la dimensione nominale della parte ed è generalmente una potenza di 2; la dimensione della parte è tipicamente scelta in base all'ammontare totale dei dati nel torrent, vincolata dal fatto che dimensioni di parti troppo grandi causano inefficienza e dimensioni di parti troppo piccole faranno aumentare la dimensione del file.torrent; la convenzione utilizzata per prendere la parte più piccola in un torrent è non più grande di 50-75kB, comunque, ora che le capacità di memorizzazione e di banda non sono troppo ristrette è meglio scegliere parti di 512 kB o inferiori; le dimensioni più comuni sono 256 kB, 512 kB e 1 MB; ogni parte è uguale agli altri ad eccezione dell'ultimo che è irregolare (intero);
- *pieces*: stringa ottenuta dalla concatenazione di tutti i valori di 20 byte hash SHA1, uno per parte (stringa);
- *private*: (opzionale) se è settato a '1', il client deve rendere nota la sua presenza per ottenere altri peer soltanto attraverso i tracker esplicitamente descritti nel file di metainformazioni; se settato a '0' o se non presente, il client può ottenere peer anche in altri modi, ad esempio DHT (intero).

Le altre informazioni contenute nel dizionario si differenziano a seconda dei casi, file singolo o multipli.

Modo singolo file

- *name*: nome del file. E' solamente per conoscenza (stringa);
- *length*: lunghezza del file in byte (intero);
- *md5sum*: (opzionale) stringa costituita da 32 caratteri esadecimali corrispondente alla somma MD5 del file (stringa).

Modo multipli file

- *name*: nome della directory nella quale si trovano tutti i file (stringa);
- *files*: lista di dizionari, uno per ogni file. Ogni dizionario in questa lista contiene le seguenti chiavi:
 - *length*: lunghezza del file in byte (intero);
 - *md5sum*: (opzionale) stringa costituita da 32 caratteri esadecimali corrispondente alla somma MD5 del file (stringa);
 - *path*: lista contenente una o più stringhe che insieme rappresentano il percorso ed il nome del file. Ogni elemento della lista corrisponde o ad un nome di directory o (nel caso in cui sia l'ultimo elemento) al nome del file. Ad esempio, il percorso '*dir1/dir2/file.ext*' consisterebbe di 3 elementi di tipo stringa '*dir1*', '*dir2*', e '*file.ext*' e sarebbe codificato come *l4:dir14:dir28:file.exte*.

PROTOCOLLO DEL TRACKER HTTP/HTTPS

Il tracker è un servizio HTTP/HTTPS che risponde a richieste HTTP GET, dette announce URL. Le richieste includono metriche del client che aiutano il tracker a fare statistiche relativamente al torrent. La risposta include una lista di peer che aiuta il client a partecipare al torrent. L'URL di base consiste nell'URL del tracker come definita nel file di metainformazioni e parametri aggiunti a tale URL usando i metodi CGI standard, ad esempio `www.tracker.bt?param1=val1¶m2=val2`. Tutti i valori binari, cioè tutti i byte non contenuti nei range 0-9, a-z, A-Z, '.', '-', '_' e '~' devono essere codificati con le giuste sequenze di escape, cioè il formato '%nn' con nn il valore esadecimale del byte.

Richiesta del client al tracker

La richiesta HTTP GET che il client invia al tracker contiene i seguenti parametri:

- *info_hash*: 20 byte SHA1 hash del valore della chiave info del file di metainformazioni; notare che questo valore è a sua volta un dizionario codificato Bencode;
- *peer_id*: stringa di 20 byte usata come ID unico per il client, generate dal client stesso allo startup. Può essere un valore qualsiasi, anche binario. Attualmente non ci sono linee guida per generare questo peer ID, comunque, si può giustamente presumere che esso debba essere unico almeno sulla macchina locale, poi potrebbe probabilmente incorporare elementi quali il process ID ed un timestamp registrato allo startup;
- *port*: numero di porta sulla quale il client è in ascolto; le porte riservate a BitTorrent sono tipicamente 6881-6889. I client possono scegliere anche numerazioni superiori se non possono sceglierne una appartenente al range;
- *uploaded*: totale dell'ammontare fornito in upload (dal momento in cui il client invia un evento 'started' al tracker) in base 10. Dal momento che non è esplicitamente detto nelle specifiche ufficiali, dovrebbe essere il numero totale di byte forniti in upload;
- *downloaded*: totale dell'ammontare fornito in download (dal momento in cui il client invia un evento 'started' al tracker) in base 10. Dal momento che non è esplicitamente detto nelle specifiche ufficiali, dovrebbe essere il numero totale di byte forniti in download;
- *left*: numero di byte che il client deve ancora scaricare, codificato in base 10;
- *compact*: settato a '1' indica che il client accetta una risposta compatta; in tal caso la lista dei peer è rimpiazzata da una stringa di peer, con 6 byte per ogni peer; I primi 4 byte sono l'host e gli ultimi 2 byte la porta. Da notare che alcuni tracker supportano soltanto risposte compatte per il risparmio di banda e altri rifiutano richieste senza 'compact=1' o semplicemente inviano una risposta compatta se la richiesta non contiene 'compact=0' (nel qual caso rifiuteranno la richiesta);
- *no_peer_id*: indica che il tracker può omettere il campo peer_id nel campo dizionario dei peer; questa opzione è ignorata se 'compact=1';
- *event*: se non specificato, la richiesta è fatta ad intervalli regolari; se specificato, deve essere un valore tra 'started', 'completed', 'stopped', (vuoto è uguale a non specificato):

- *started*: la prima richiesta al tracker deve includere la chiave event con questo valore;
 - *stopped*: deve essere inviato al tracker se il client si sta spegnendo regolarmente;
 - *completed*: deve essere inviato al tracker quando il download è completato, ma non se il download era già completo allo startup del client; presumibilmente è per permettere al tracker di incrementare la metrica 'completed downloads' basata unicamente su event;
- *ip*: (opzionale) il vero indirizzo IP della macchina client in notazione decimale puntata o l'indirizzo IPv6 in notazione esadecimale come definito nelle rfc3513; in generale questo parametro non è necessario dato che l'IP del client può essere determinato dall'indirizzo IP da cui è arrivata la richiesta HTTP; il parametro è necessario solo nel caso in cui l'indirizzo IP riportato nella richiesta non sia l'indirizzo IP del client, ad esempio quando il client comunica col tracker attraverso un proxy o quando entrambi (client e tracker) sono dallo stesso lato locale di un NAT gateway. La ragione d'esistere di tale chiave è inoltre il fatto che il tracker vorrebbe eliminare l'indirizzo interno del client che non è routeable; inoltre, il client deve esplicitamente mostrare il suo esterno e routeable indirizzo IP per essere contattato da peer esterni. I vari tracker trattano questo parametro diversamente: alcuni ne tengono conto soltanto se l'indirizzo IP riportato nella richiesta è nello spazio delle RFC1918; altri ne tengono conto incondizionatamente, mentre altri lo ignorano completamente; nel caso di indirizzo IPv6 (es: 2001:db8:1:2::100) esso indica solo che il client può comunicare tramite IPv6;
- *numwant*: (opzionale) numero di peer che il client vorrebbe ricevere dal tracker; può anche essere 0 e se omissso, il tipico valore di default è 50 peer;
- *key*: (opzionale) identificazione addizionale che non è scambiata con altri utenti; dovrebbe aver lo scopo di permettere ai client di provare la loro identità se il loro indirizzo IP cambiasse;
- *trackerid*: (opzionale) se una richiesta precedente contenesse un tracker ID questa chiave sarebbe settata con tale valore.

Risposta del tracker al client

La risposta che il tracker invia al client è un testo in chiaro che consiste in un dizionario codificato Bencode con le seguenti chiavi:

- *failure reason*: se presente, nessun'altra chiave dovrebbe essere presente; il valore è un messaggio di errore umanamente comprensibile relativo al motivo del fallimento della richiesta (stringa).
- *warning message*: (nuovo, opzionale) simile al failure reason, ma la risposta può essere processata normalmente perché è un errore meno grave (stringa);
- *interval*: intervallo di tempo, in secondi, che il client deve attendere tra un invio di una richiesta e l'altro al tracker (intero);
- *min interval*: (opzionale) intervallo di tempo minimo tra richieste; se presente, i client non devono inviare richieste con una frequenza superiore a tale soglia (intero);
- *tracker id*: stringa che il client riporta nella successiva richiesta al tracker; se assente e una richiesta precedente riportava un tracker id, il client utilizza il vecchio valore (stringa);
- *complete*: numero di peer con l'intero file, cioè i seeder (intero);
- *incomplete*: numero di peer non seeder, cioè leecher (intero);
- *peers*: lista dei peer fornita dal tracker per far partecipare il client al torrent; può seguire due modelli:

- (modello dizionario) il valore è una lista di dizionari, ognuno con le seguenti chiavi:

- *peer id*: ID che il peer si è auto assegnato (stringa);
- *ip*: indirizzo IP del peer o IPv6 (esadecimale) o IPv4 (decimale puntato) o nome DNS (stringa);
- *port*: numero di porta che il peer utilizza (intero);

- (modello binario) il valore è una stringa multipla di 6 byte, i primi 4 sono l'indirizzo IP e gli ultimi 2 byte sono il numero di porta.

Come già detto, la lista dei peer di default contiene 50 elementi; se ci sono meno peer la lista risulta più corta.

Il tracker sceglie in modo random i peer da includere nella risposta, ma può scegliere di implementare un meccanismo più intelligente per selezionare i peer.

I client possono inviare richieste al tracker più spesso di quanto specificato da interval, se si verifica un evento (ad es. stopped o completed) o se il client ha bisogno di conoscere altri peer; comunque è considerata cattiva pratica l'inondare il tracker di richieste per ottenere più peer; quindi se un client vuole un maggior numero di peer nella risposta del tracker, può specificarlo col parametro numwant una sola volta.

Nota per gli implementatori: già 30 peer sono un numero abbondante, la versione ufficiale del client infatti crea nuove connessioni solo se ha meno di 30 peer e rifiuta connessioni se ne ha già 55; questo valore è importante per le performance: quando una nuova parte è stata completamente scaricata, è necessario inviare messaggi HAVE al maggior numero di peer attivi, di conseguenza il costo del traffico broadcast cresce in proporzione al numero di peer; sopra 25, non cresce neanche più di tanto la velocità di download.

CONVENZIONE TRACKER SCRAPE

Per convenzione molti tracker supportano un altro tipo di richiesta, che domanda lo stato del torrent (o di tutti i torrent) che il tracker gestisce; tale richiesta è riferita come 'scrape page' dato che automatizza il noioso processo di 'screen scraping' della pagina di stato del tracker.

Richiesta scrape

La scrape URL è anch'esso un metodo HTTP GET, simile alla richiesta announce; ma l'URL è diversa; per ricavare la scrape URL si procede nel seguente modo: si prende l'announce URL, si cerca l'ultima '/' al suo interno, se il testo immediatamente successivo ad essa è 'announce', lo si sostituisce col testo 'scrape' e il tracker fornisce la scrape page; se il testo 'announce' non è presente, significa che il tracker non supporta la richiesta scrape. Ad esempio:

~http://example.com/a -> (scrape not supported)

~http://example.com/announce?x2%0644 -> ~http://example.com/scrape?x2%0644

Può essere aggiunto anche il parametro opzionale info_hash, valore di 20 byte già descritto; questo riduce il report del tracker ad un torrent in particolare, altrimenti la richiesta ritorna lo stato di tutti i torrent che il tracker gestisce; l'autore della richiesta è fortemente incoraggiato ad utilizzare tale parametro per ridurre il carico e la banda del tracker.

Si possono indicare anche parametri info_hash multipli.

Non è un parametro delle specifiche ufficiali, ma è diventato uno standard defacto.

Risposta scrape

La risposta a tale metodo HTTP GET è un testo in chiaro o talvolta un documento compresso gzip, che consiste in un dizionario codificato Bencode contenente le seguenti chiavi:

- *files*: un dizionario contenente una coppia chiave-valore per ogni torrent; se il parametro *info_hash* c'è ed ha un valore valido, il dizionario conterrà una sola coppia chiave-valore. La chiave di ogni coppia consiste in un valore *info_hash* binario di 20 byte, mentre il valore è a sua volta un altro dizionario contenente le seguenti chiavi:
 - *complete*: numero di peer con l'intero file, cioè i seeder (intero);
 - *downloaded*: numero totale di volte che il tracker ha registrato il completamento ('event=complete'), cioè in cui un client ha terminato il download di un torrent (intero);
 - *incomplete*: numero di peer non seeder, cioè i leecher (intero);
 - *name*: (opzionale) nome interno del torrent, come specificato nel campo nome nella sezione info del file.torrent (stringa).

Notare che questa risposta ha 3 livelli di dizionari annidati; ad esempio:

```
d5:filesd20:.....d8:completei5e10:downloadedi50e10:incompletei10eeee
```

Dove sono i 20 byte di *info_hash* e ci sono 5 seeder, 10 leecher e 50 download completati.

PROTOCOLLO TRA PEER (TCP)

Il protocollo tra i peer facilita lo scambio di parti.

I peer comunicano scambiando messaggi ed ogni peer mantiene informazioni di stato relative agli altri peer.

Informazioni di stato

Un client deve mantenere informazioni di stato per ogni connessione stabilita con un peer remoto; tali informazioni sono in particolare due:

- *choked*: se il peer remoto ha o no bloccato (choked) il client; quando un peer blocca il client, il client interpreta questo come la notifica che il peer remoto non risponderà ad alcuna sua richiesta finché il client non verrà sbloccato (unchoked). Il client non

dovrebbe tentare di inviare richieste per blocchi al peer remoto che lo ha bloccato e dovrebbe considerare tutte le richieste pendenti (senza risposta) come scartate dal peer remoto;

- *interested*: se il peer remoto è o no interessato a dati che il client offre; questa è la notifica che il peer remoto richiederà dei blocchi al client quando il client sbloccherà (unchoke) il peer.

Notare che questo implica anche che il client avrà bisogno di tenere traccia anche se esso stesso è o no interessato (*interested*) a contenuti di un peer remoto e se ha peer remoti bloccati (*choked*) o no (*unchoked*).

Quindi, le informazioni che il client tiene sono in realtà duplicate, cioè relative al peer nei confronti di se stesso e viceversa:

- *am_choking*: il client blocca il peer remoto;
- *am_interested*: il client è interessato al peer remoto;
- *peer_choking*: il peer blocca il client;
- *peer_interested*: il peer è interessato al client.

Tutte le connessioni del client, all'avvio sono bloccate (*choked*) e non interessate (*not interested*).

In altre parole, il settaggio dei vari parametri di stato è il seguente:

- *am_choking* = 1, cioè il client ha bloccato il peer remoto;
- *am_interested* = 0, cioè il client non è interessato al peer remoto;
- *peer_choking* = 1, cioè il peer remoto ha bloccato il client;
- *peer_interested* = 0, cioè il peer remoto non è interessato al client.

Il client scarica un blocco (download) quando il client è interessato ad un peer remoto (*am_interested* = 1) e quel peer remoto non ha bloccato il client (*peer_choking* = 0).

Il client fornisce un blocco (upload) quando il client non ha bloccato il peer remoto (*am_choking* = 0) ed quel peer remoto è interessato al client (*peer_interested* = 1).

E' importante che il client informi i suoi peer se è o no interessato (*interested*) a loro; questa informazione di stato dovrebbe essere scambiata con ogni peer ogni volta che questo blocca (*choked*) il client; questo permetterebbe al peer remoto di sapere se quando sbloccherà (*unchoked*) il client, questo farà download dal peer oppure no (e viceversa).

Tipi di dato

Senza ulteriori specifiche, tutti gli interi nel protocollo tra peer sono codificati come valori big-endian a 4 byte.

Flusso di messaggi

Il protocollo tra peer consiste in un messaggio iniziale detto handshake; dopodiché i peer comunicano attraverso lo scambio di messaggi di lunghezza prefissata espressa con un intero.

Handshake

L'handshake è un messaggio obbligatorio e deve essere il primo messaggio trasmesso dal client, per aprire la comunicazione con un peer remoto.

Ogni peer invia ai suoi peer un messaggio handshake e ne riceve uno in risposta.

Il messaggio handshake è così composto:

handshake: <ptrlen><ptr><reserved><info_hash><peer_id>

- *ptrlen*: 1 byte che esprime la lunghezza della stringa <ptr>;
- *ptr*: stringa identificativa del protocollo;
- *reserved*: 8 byte riservati; tutte le implementazioni correnti usano tutti 0; ogni bit in questi byte potrebbe essere utilizzato per cambiare il comportamento del protocollo;
- *info_hash*: 20 byte di hash SHA1 della chiave info nel file di metainformazioni; è lo stesso info_hash trasmesso nelle richieste al tracker;
- *peer_id*: 20 byte, stringa che esprime un ID unico del client; generalmente è lo stesso peer_id trasmesso nelle richieste al tracker (ma non sempre, ad esempio non quando viene usata l'opzione 'anonimità' nel client Azureus).

Ha una dimensione di:

1 byte <ptrlen> + lunghezza <ptr> + 8 byte <reserved> + 20 byte <info_hash> + 20 byte <peer_id> = (49+len(ptr)) byte.

Nella versione 1.0 del protocollo BitTorrent *ptrlen* = 19, and *ptr* = 'BitTorrent protocol'.

L'iniziatore di una connessione deve trasmettere l'handshake immediatamente; il ricevente può attendere l'handshake dell'iniziatore della connessione se esso ha la capacità di servire multipli torrent contemporaneamente (i torrent sono univocamente identificati dal loro info_hash); comunque, il ricevente deve rispondere prima possibile una volta ricevuta la parte info_hash dell'handshake.

Se un client riceve un handshake con una info_hash che non sta attualmente servendo, il client deve lasciar cadere la connessione.

Se l'iniziatore di una connessione riceve un handshake nel quale il peer_id non è quello che si aspetta, lascia cadere la connessione; notare che un iniziatore riceve presumibilmente le informazioni del peer dal tracker, il quale include il peer_id scelto dal peer stesso; il peer_id proveniente dal tracker ed il peer_id contenuto nell'handshake ci si aspetta che siano uguali.

Peer_ID

Il peer_id ha una dimensione di esattamente 20 byte (caratteri).

Esistono principalmente 2 convenzioni su come codificare il client e la versione del client in un peer_id:

- Stile Azureus;
- Stile Shadow's.

Stile Azureus

Lo stile Azureus utilizza la seguente codifica:

<-><2 caratteri per il client ID><4 cifre ASCII per il numero di versione><-><numeri random>.

Per esempio: '-AZ2060-' per Azureus 2.0.6.0 con '-', AZ per Azureus, 2060 per la versione e '-'.

Tra i client noti che utilizzano questa codifica, con i relativi 2 caratteri di client ID, ci sono: Azureus (AZ), BitComet (BC), DelugeTorrent (DE), Halite (HL), Ktorrent (KT), libtorrent (LT), libTorrent (lt), qBittorrent (qB), Transmission (TR).

Stile Shadow's

Lo stile Shadow's utilizza la seguente codifica:

<1 carattere ASCII alfanumerico come client ID><fino a 5 caratteri per il numero di versione (con '-' come carattere di padding se sono meno di 5)><3 caratteri (di solito '---', ma non sempre)>< caratteri random>.

Ogni carattere nella stringa che rappresenta il numero di versione è un numero da 0 a 63: '0'=0, ..., '9'=9, 'A'=10, ..., 'Z'=35, 'a'=36, ..., 'z'=61, '='=62, '-'=63.

Per esempio: 'S58B-----' sta per Shadow's 5.8.11 con S per Shadow's come client ID, 58B-- come numero di versione, seguiti da '---'.

Tra i client noti che utilizzano questa codifica, con i relativi 2 caratteri di client ID, ci sono: BTQueue (Q), BitTornado (T), UPnP NAT Bit Torrent (U).

Messaggi

Tutti gli altri messaggi del protocollo hanno il formato seguente:

<length prefix><message ID><payload>.

Il length prefix è un valore codificato big-endian di quattro byte.

Il message ID è un carattere decimale.

Il payload dipende dal tipo di messaggio.

I messaggi sono:

- keep-alive: *<len=0000>.*

Il messaggio keep-alive è un messaggio di zero byte, come specificato dal length prefix settato a zero. Non è presente message ID né payload. I peer se non ricevono più messaggi (keep-alive o altri messaggi) per un certo periodo di tempo, possono chiudere una connessione; quindi vengono inviati messaggi keep-alive per mantenere viva la connessione se per un certo intervallo di tempo, tipicamente della durata di due minuti, non viene inviato alcun messaggio.

- choke: *<len=0001><id=0>.*

Il messaggio choke ha lunghezza fissa ed è privo di payload.

- unchoke: *<len=0001><id=1>.*

Il messaggio unchoke ha lunghezza fissa ed è privo di payload.

- interested: *<len=0001><id=2>.*

Il messaggio interested ha lunghezza fissa ed è privo di payload.

- not interested: *<len=0001><id=3>.*

Il messaggio not interested ha lunghezza fissa ed è privo di payload.

- have: *<len=0005><id=4><piece index>.*

Il messaggio have ha lunghezza fissa. Il payload è l'indice della parte che è stata appena scaricata (downloaded) con successo e verificata tramite l'hash.

Nota per gli implementatori: dato che i peer non vogliono scaricare parti che hanno già, un peer può scegliere di non comunicare che ha una parte ad un peer che già ce l'ha. Risulterebbe così una riduzione del numero di messaggi have almeno del 50 %, riducendo così l'overhead del protocollo di circa il 25 % - 35 %. Allo stesso tempo può

essere utile inviare un messaggio have ad un peer che ha già quella parte, come informazione che poi gli sarà utile per la determinazione delle parti rare.

Un peer malizioso potrebbe anche scegliere di annunciare parti con messaggi have sapendo che un peer non scaricherà mai; quindi modellare i peer usando questa informazione non è una buona idea.

- bitfield: $\langle len=0001+X \rangle \langle id=5 \rangle \langle bitfield \rangle$.

Il messaggio bitfield può essere inviato solo immediatamente dopo che una sequenza di handshaking è completata e prima che altri messaggi siano inviati; e' opzionale e non è necessario inviarlo se un client non ha parti.

Il messaggio bitfield ha lunghezza variabile, dove X rappresenta la lunghezza del campo bitfield. Il payload è un campo di bit che rappresenta le parti che sono stati scaricati con successo. Il primo bit, il più alto, nel primo byte rappresenta la parte di indice 0; i bit non settati indicano che quello che rappresentano è una parte mancante, mentre quelli settati indicano una parte valida e disponibile per essere scaricata; i bit rimanenti sono settati a zero.

Un bitfield di lunghezza errata è considerato un errore; i client possono chiudere la connessione se ricevono bitfield che non hanno la dimensione corretta o se il bitfield ha alcuni dei bit in più settato e non a zero.

- request: $\langle len=0013 \rangle \langle id=6 \rangle \langle index \rangle \langle begin \rangle \langle length \rangle$.

Il messaggio request è di lunghezza fissa ed è utilizzato per richiedere blocchi. Il payload contiene le seguenti informazioni:

- index: indice della parte (intero);
- begin: offset di inizio blocco all'interno della parte (intero);
- length: lunghezza del blocco richiesto (intero).

- piece: $\langle len=0009+X \rangle \langle id=7 \rangle \langle index \rangle \langle begin \rangle \langle block \rangle$.

Il messaggio piece è di lunghezza variabile, dove X è la dimensione del blocco.

Il payload contiene le seguenti informazioni:

- index: indice della parte (intero);
- begin: offset di inizio blocco all'interno della parte (intero);
- block: blocco di dati che è un sottoinsieme della parte specificata da index.

- cancel: `<len=0013><id=8><index><begin><length>`.

Il messaggio cancel è di lunghezza fissa ed è utilizzato per cancellare richieste di blocchi; il payload è identico a quello del messaggio request. Questo messaggio è tipicamente utilizzato durante la fase End Game.

- port: `<len=0003><id=9><listen-port>`.

Il messaggio port è utilizzato nelle nuove versioni di Mainline che implementano il DHT tracker. E' un messaggio di lunghezza fissa e la listen port è la porta su cui il nodo DHT del peer resta in ascolto; il peer dovrebbe essere inserito nella tabella di routing locale se il DHT tracker è supportato.

ALGORITMI

Strategia di download

I client scelgono le parti da scaricare in modo random. La miglior strategia è scaricare le parti a partire dalla più rara, seguendo l'algoritmo Rarest First.

Il client determina quale sia la parte più rara tramite i messaggi bitfield ricevuti da ogni peer, ed aggiorna questi ultimi tramite l'invio di messaggi have. Poi, il client può scaricare le parti che compaiono meno frequentemente nei bitfield giunti dai peer.

End Game

Quando un download è quasi completato, c'è la tendenza, per gli ultimo pochi blocchi, a calare lentamente; per aumentare la velocità, il client invia richieste per tutti i suoi blocchi mancanti a tutti i suoi peer e per evitare un overhead eccessivo che porterebbe al degrado delle prestazioni, il client invia anche messaggi cancel per un certo blocco a tutti i peer ogni volta che riceve quel blocco.

Non ci sono soglie documentate, percentuali raccomandate o numero di blocchi utilizzato come guida per decidere quale sia il momento giusto di avviare l'algoritmo end game, è un argomento ancora in fase di discussione. Alcuni client entrano in modalità end game quando tutte le parti sono stati richiesti, altri attendono finché il numero di blocchi mancanti è minore del numero di blocchi in transito e non superiore a 20.

L'idea dovrebbe essere quella di attivare la modalità end game quando il numero di blocchi mancanti è piccolo, in modo da ridurre l'overhead e randomizzare le richieste dei blocchi per abbassare la possibilità di scaricare duplicati.

Choking and Optimistic Unchoking

Il choking è utilizzato per molte ragioni. Il meccanismo di controllo della congestione di TCP si comporta in modo non molto soddisfacente quando gestisce più connessioni alla volta. Inoltre il choking permette ad ogni peer di utilizzare un algoritmo tit-for-tat per assicurare che essi ottengano un consistente download rate.

Esistono vari criteri per creare un buon algoritmo di choking. Esso dovrebbe fissare un tetto per il numero di simultanei upload in modo da ottenere buone prestazioni del TCP; dovrebbe evitare la cosiddetta 'fibrillazione', cioè l'alternarsi di choking ed unchoking con un'elevata frequenza; dovrebbe garantire reciprocità tra i peer ed infine dovrebbe implementare il cosiddetto optimistic unchoking, cioè chiudere connessioni non utilizzate in modo da rimpiazzare connessioni migliori con quelle correntemente utilizzate.

L'attuale implementazione dell'algoritmo di choking evita la fibrillazione variando i peer bloccati (choked) ogni dieci secondi.

La reciprocità ed il tetto massimo di upload consentiti sono gestiti sbloccando (unchoked) i quattro peer che hanno il miglior upload rate e che sono interessati (interested). Questo massimizza il download rate del client. Questi quattro peer sono chiamati downloader dato che sono interessati a scaricare dal client.

I peer che hanno il miglior upload rate, ma non sono interessati (not interested) vengono sbloccati (unchoked); se diventano interessati (interested), il downloader con il peggior upload rate viene bloccato (choked); se un client ha un file completo, utilizza il suo upload rate piuttosto che il suo download rate per decidere quali peer sbloccare (unchoked).

Grazie all'optimistic unchoking, ogni volta c'è un singolo peer che è sbloccato (unchoked) malgrado il suo upload rate (se interessato, esso conta come uno che quattro downloader). I peer sbloccati da tale algoritmo ruotano ogni 30 secondi. I peer che stabiliscono nuove connessioni riescono a trovare così una prima parte da scaricare e fornire poi in upload.

Anti-snubbing

Occasionalmente un peer BitTorrent può essere bloccato (choked) da tutti i peer dai quali sta facendo download; in tal caso esso continuerà a ottenere bassi download rate finché l'optimistic unchoke trova peer migliori. Per mitigare questo problema, ogni volta che il client non ottiene alcuna parte da un peer da cui sta scaricando per più di 1 minuto, assume

di esser stato snobbato (snubbed) dal peer e non fa più upload ad esso eccetto nel caso venga selezionato dall'optimistic unchoke.

ESTENSIONI UFFICIALI DEL PROTOCOLLO

Attualmente esistono poche estensioni ufficiali del protocollo.

Fast Peers Extensions

- Bit Riservato: il terzo bit meno significativo nell'ottavo byte del campo reserved, cioè $\text{reserved}[7] = 0x04$.

Queste estensioni servono per ragioni multiple. Permettono ad un peer di fare velocemente il bootstrap in uno swarm dando ad un peer uno specifico set di parti che potranno essere scaricati senza badare allo stato se bloccato (choked); riducono i messaggi di overhead aggiungendo messaggi HaveAll e HaveNone e permettono il rifiuto esplicito di richieste di parti mentre prima era possibile soltanto un rifiuto implicito lasciando che un peer aspettasse una parte che non sarebbe mai arrivata.

Distributed Hash Table

- Bit riservato: l'ultimo bit significativo nell'ottavo byte del campo reserve, cioè $\text{reserved}[7] = 0x01$.

Questa estensione serve per permettere di tracciare i peer che scaricano i torrent senza l'uso di un tracker standard; un peer che implementa tale protocollo diventa a sua volta un tracker e memorizza liste di altri nodi peer che possono essere utilizzati per trovare nuovi peer.

Consente di rendere il protocollo più decentralizzato.

Connection Encryption

Questa estensione permette la creazione di connessioni criptate tra peer che possono essere utilizzate per bypassare ISP che strozzano il traffico BitTorrent.

Ringraziamenti

A conclusione del mio percorso universitario vorrei dedicare questo spazio ad un bilancio e alle persone che mi sono state vicine in questo cammino.

Dal punto di vista formativo, il mio percorso di dottorato ha arricchito le mie conoscenze e il mio bagaglio di esperienze personali. Ho imparato molto riguardo all'approccio scientifico di ricerca e a tecniche di risoluzione dei problemi, ho potuto apprendere linguaggi di programmazione che non conoscevo, ed ho potuto spaziare anche in aree di interesse non diretto, ed ho potuto approfondire l'apprendimento della lingua inglese. Ho avuto inoltre la possibilità di partecipare a incontri e conferenze internazionali in cui mi sono confrontata con grandi personalità nell'ambito della ricerca. Ed ho avuto anche la possibilità di fare le esperienze didattiche che tanto amo.

Inoltre, in particolare l'interesse verso il mio argomento di ricerca, cioè l'efficienza energetica, mi ha permesso di sviluppare sensibilità verso il problema energetico ed una coscienza verso la sostenibilità che considero ormai parte di me e che ho fatto mia come una sorta di missione. La possibilità infatti di vivere in un ambiente sostenibile migliorerebbe la vita di tutta la nostra società e consentirebbe il rispetto e maggior longevità dell'ambiente in cui viviamo.

Dal punto di vista umano e dei legami personali, vorrei ringraziare coloro che ci sono stati.

Mio marito Marco per la sua presenza e la sua pazienza, e per il sostegno e l'amore che mi ha fatto sentire anche nell'affrontare questi tre anni di dottorato.

La mia famiglia, sempre vicina, ed in particolare la mia mamma che mi spinge sempre oltre i miei limiti.

Il Prof. Anastasi, senza il quale questo percorso non sarebbe stato così ricco, che mi ha offerto dapprima la possibilità del dottorato e che mi ha poi supportato nella ricerca e in tutte le esperienze ad essa correlate con la sua disponibilità e con il suo tempo, confermando ancora una volta, dopo entrambe le mie tesi di laurea (triennale e specialistica), e due esami, la mia stima come docente, come tutor e come persona.

A Totta, l'amica, la testimone, la personal trainer, la motivatrice che è, che ha saputo sdrammatizzare i miei deliri, che mi ha dato dato fiducia quando non pensavo di farcela, che ha riso con me per i risultati raggiunti.

Alla passione, che mi fa affrontare le esperienze della vita con la pancia, e all'ottimismo.