Strumenti della Teoria dei Giochi per l'Informatica **A.A. 2009/10**

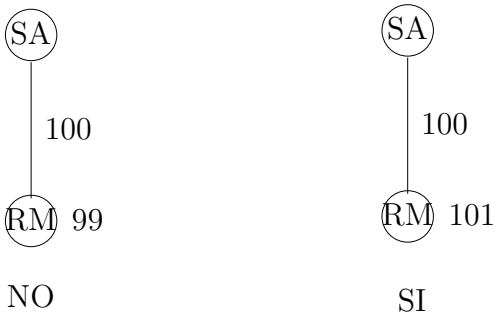
Lecture 11: 13-14 Maggio 2010

Meccanismi per la Condivisione dei Costi

Docente *Paolo Penna* Note redatte da: *Paolo Penna*

1 Primo Esempio

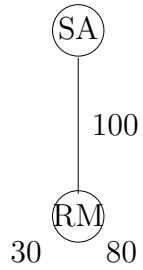
Vogliamo vendere un oggetto, ed essere sicuri che chi compra ci paga (almeno) il costo della spedizione:



Asta 2ndo Prezzo (Vickrey): Una sola offerta $\Longrightarrow$ Il vincitore **non paga nulla!**

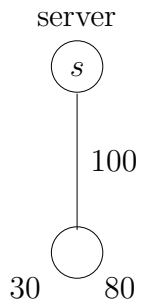
Asta con Prezzo di Riserva: Il veditore partecipa all'asta

Esercizio: Considera questo esempio (due offerte):



L'Asta di 2ndo Prezzo copre il costo? Cosa fa l'Asta con Prezzo di Riserva?

Un problema **quasi** uguale



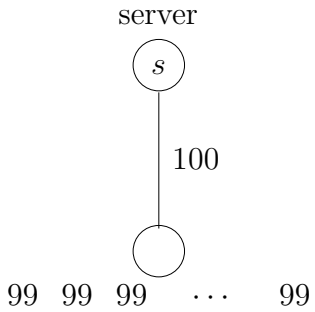
100 = costo
se **uno** o **entrambi**
ricevono

Insieme possono pagare il costo
Asta con Prezzo di Riserva $\Longrightarrow$ Nessuno servito

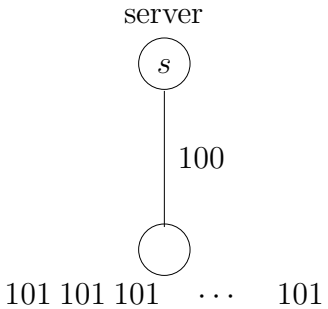
Benessere Sociale = Contentezza collettiva − Costo collettivo (1)

$30 + 80 - 100$ è meglio di 0

Asta 2ndo Prezzo $\Longrightarrow$ Ignora il costo
Asta Prezzo di Riserva $\Longrightarrow$ Ignora Contentezza collettiva (uno solo alla volta)

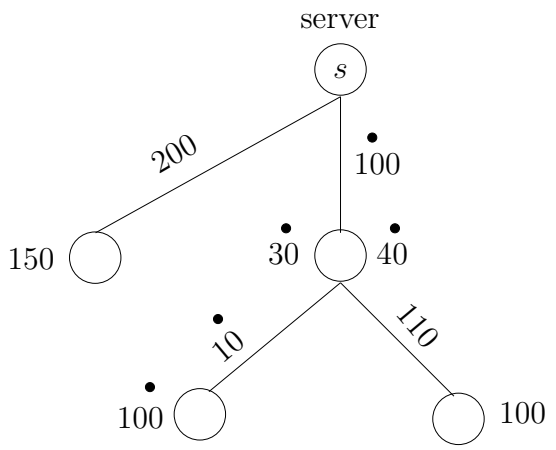


nessuno



uno solo

2 Multicast su Alberi



costo = somma link usati

benessere sociale = 100 + 30 + 40 - (100 + 10)

Problema (Multicast su Albero)
Dato un albero con costi (noti) sui link, decido quali nodi raggiungere per massimizzare il benessere sociale (in base alle valutazioni degli agenti).

Usiamo uno schema (VCG) simile allo shortest-path della lezione precedente:

MASSIMO BENESSERE SOCIALE SENZA i — BENESSERE SOCIALE OTTIMO MA NON CONTANDO i

Senza dire esattamente come sono fatte queste due quantità, possiamo dire che i pagamenti avranno la forma

$$P_i^{BS}(v) = BS_{-i}(v_{-i}) - BS_{v_i=0}(v)$$

Funziona?
(compatibile agli incentivi)

Arriviamo a un meccanismo VCG

Input: Valutazioni $v_1(), \ldots, v_n()$
Algoritmo: Trova la solutione S^* che massimizza la somma delle valutazioni: $v_1(S) + \cdots + v_n(S)$
Pagamenti: Agente i paga

$$P_i^{VCG}(c) = h_i(v_{-i}) - \sum_{j \neq i} v_j(S^*)$$

dove $h_i()$ non dipende da v_i.

Nel nostro caso (multicast su albero):

S = chi riceve la trasmissione (servito) = soluzione

$C(S)$ = costo (minimo) per trasmettere ad S

$BS(S, v) = \left(\sum_{i \in S} v_i\right) - C(S)$ = benessere sociale

$BS(v) = \max_S BS(S, v)$ = benessere sociale ottimo (massimo)

Il benessere sociale **non è la somma** delle valutazioni:

$$v_1(S) + \cdots + v_n(S) = \sum_{i \in S} v_i \neq BS(S, v) = \sum_{i \in S} v_i - C(S)$$

Esercizio: Se applichiamo VCG a queste valutazioni, quale soluzione calcola il meccanismo (algoritmo di VCG) nell'esempio a pagina 3?

Trucchetto: Il server "partecipa all'asta" (contiamo anche il costo)

Aggiungiamo l'agente "speciale" 0

$$v_0(S) = -C(S)$$

che fa si che la somma delle valutazioni ci dia proprio il benessere sociale:

$$v_1(S) + \cdots + v_n(S) + v_0(S) = \sum_{i \in S} v_i - C(S) = BS(S, v)$$

Algoritmo: Ora VCG calcola la soluzione S^* col benessere sociale ottimo.
Pagamenti: Traduciamo in formule l'intuizione iniziale:

$BS_{-i}(v_{-i})$ = massimo benessere sociale senza $i = \max_S \left(\sum_{j \neq i} v_j(S)\right)$

$BS_{v_i=0}(v_{-i})$ = benessere sociale ottima senza contare $i = \sum_{j \neq i} v_j(S^*)$

e la "magia" è fatta: i pagamenti P^{BS} sono della forma P^{VCG} (confronta le due formule a pagina 3).

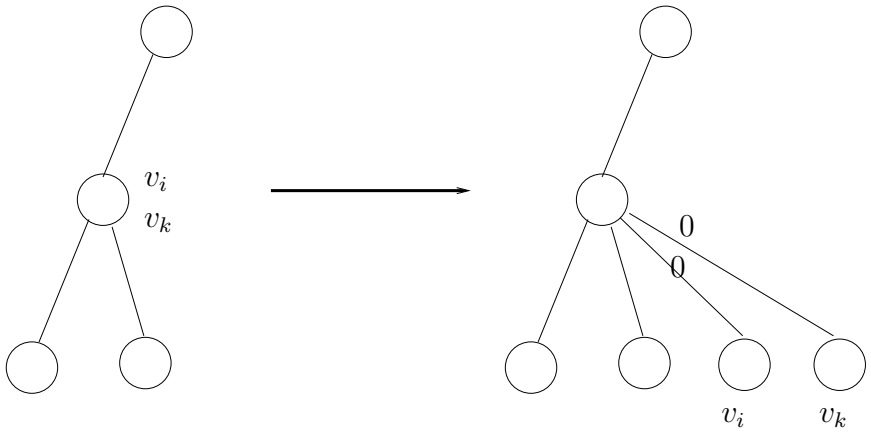
Teorema: Il problema del multicast su albero ha un meccanismo (A, P^{BS}) compatibile agli incentivi.
Dimostrazione: L'algoritmo sopra fa la stessa cosa dell'algoritmo di VCG applicato alle valutazioni con l'aggiunta dell'agente "speciale" $(v_0(S) + v_1(S) + \cdots +_n (S))$. I pagamenti P^{BS} rientrano nella definizione P^{VCG} e quindi il meccanismo risultante è compatibile agli incentivi.

2.1 Meccanismo Distribuito

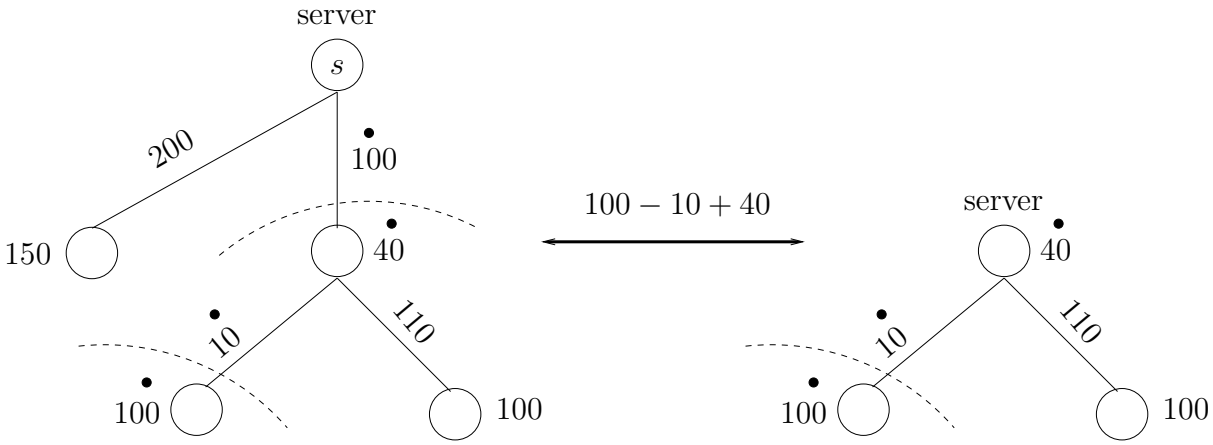
Ogni nodo decide

1. Quali figli vengono raggiunti
2. I pagamenti degli agenti su questo nodo

Una piccola semplificazione: ogni nodo ha un solo agente



Idea: calcoliamo l'ottimo dei sottoalberi



Idea: Ci conviene pagare 100 solo se poi il sottoalbero ci "ripaga" di questo costo
Ogni nodo si calcola il benessere sociale ottimo del suo sottoalbero con questo algoritmo:

Algoritmo nodo i

1. $BS^{(j)} \longleftarrow v_j$
2. Per ogni figlio j di i fai le cose seguenti:
   (a) Ricevi $BS^{(j)}$ dal nodo j
   (b) Se $BS^{(j)} \geq c_j$ allora poni $BS^{(j)} \longleftarrow BS^{(i)} + BS^{(j)} - c_j$
3. Invia $BS^{(i)}$ al padre di i

Il calcolo avviene dalle foglie (che eseguono solo il passo 1). Si può dimostrare[1] che $BS^{(i)}$ è il benessere sociale ottimo dell'istanza in cui la rete è il sottoalbero con i come radice (ossia

[1] Saltiamo questa dimostrazione

il server è i). Una volta arrivati al nodo s (server) abbiamo il benessere sociale ottimo che cercavamo.

Quanto costa il calcolo?

Ogni arco viene attraversato da esattamente un messaggio da "padre a figlio" e quindi in totale abbiamo $n-1$ messaggi. Possiamo modificare l'algoritmo per calcolare la soluzione (i invia un messaggio ai figli se scelgo l'arco c_j) e arriviamo a $2n-2$ messaggi.

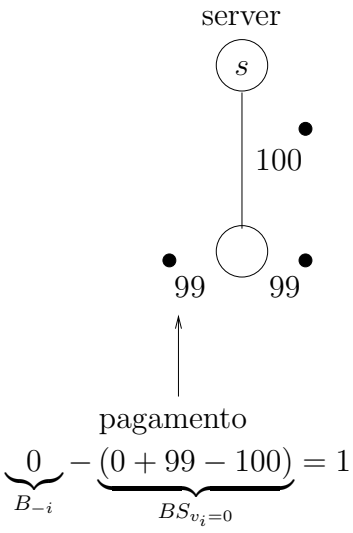
E come calcolo i pagamenti?

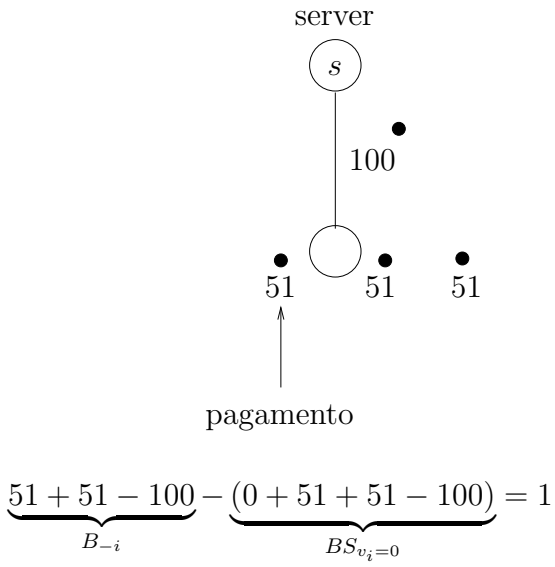
Per calcolare il pagamento di i basta calcolare $BS_{-i}(v)$ e $BS_{v_i=0}(v)$. **Idea:** faccio rigirare l'algoritmo. Per $BS_{-i}(v)$ basta far girare l'algoritmo con $v_i = 0$. Mentre $BS_{v_i=0}(v)$ lo posso calcolare direttamente dall'ottimo $BS(v)$: se i è servito nell'ottimo allora $BS_{v_i=0}(v) = BS(v) - v_i$; altrimenti, se i non è servito nell'ottimo, allora $BS_{v_i=0}(v) = BS(v)$. Per calcolare P_i spendiamo altri $2n-2$ messaggi.

Abbiamo ottenuto un meccanismo distribuito compatibile agli incentivi per il problema del multicast su alberi. Il meccanismo calcola la soluzione ottima e i pagamenti utilizzando al più $2n-2+n(2n-2)$ messaggi in totale.

2.2 Meccanismi "statali"

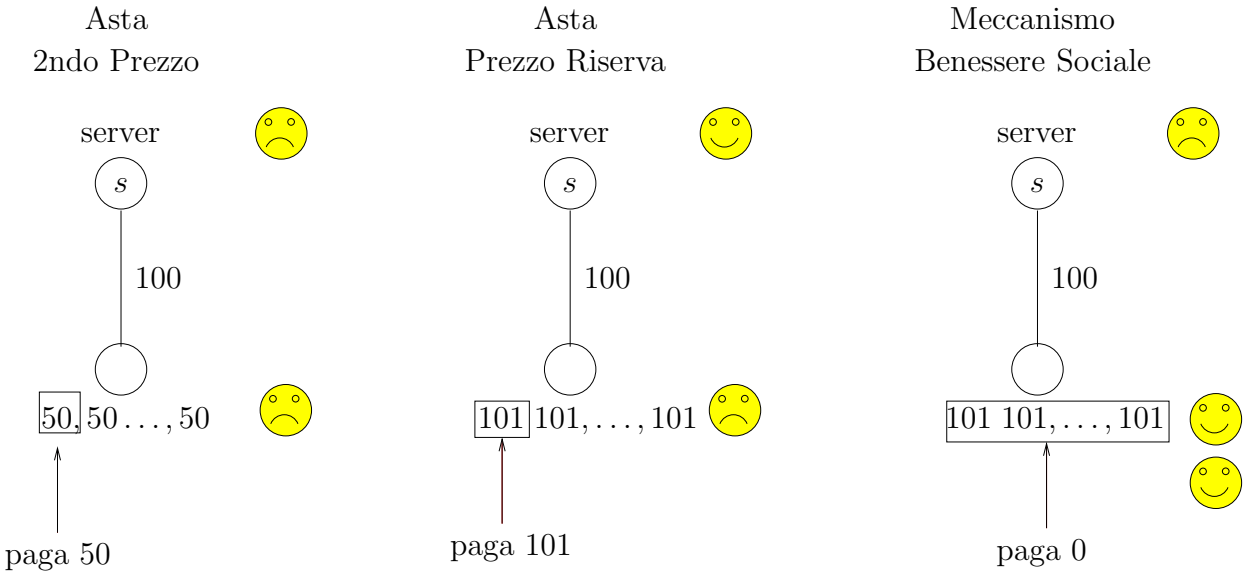
Quanto riceve il server se usa questi meccanismi (benessere sociale)?





Se il servizio è molto importante per gli agenti, il meccanismo (stato) lo fornisce (quasi) gratis

3 Riassumiamo la situazione



Posso fare entrambi contenti?

server = non ci rimetto

utenti = non paghiamo più del necessario

Bilancio in Pareggio (Budget-Balance)

$$C(S) \leq \sum_{i \in S} P_i(v) \leq C(S) \tag{2}$$

Versione approssimata:

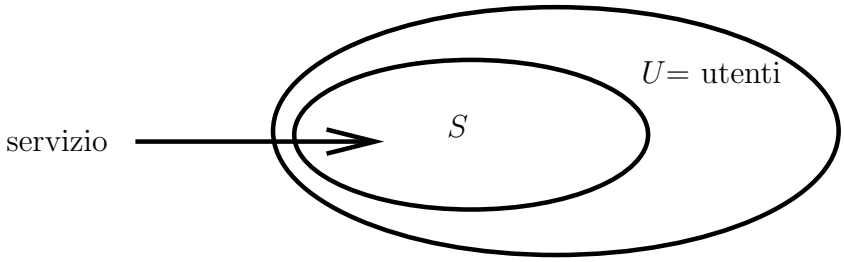
$$C(S) \leq \sum_{i \in S} P_i(v) \leq \alpha C(S) \quad \alpha \geq 1$$

(α piccolo $\iff$ nessuno (server, utenti) troppo scontento)

Esistono meccanismi compatibili agli incentivi e budget-balance?

4 Meccanismi Budget-Balance

Vediamo la cosa in modo più astratto:



1) Chi riceve il servizio?
2) Quanto paga?

Dipende dal costo $C(S)$ e da quanto gli utenti vogliono pagare

Due meccanismi **scemi:**

1. Tutti serviti, tutti devono pagare la stessa frazione del costo (ossia $C(U)/|U|$).
2. Nessuno servito, nessuno paga

Tutti e due sono budget-balance[2] e compatibili agli incentivi (qualsiasi cosa dichiarano gli utenti/agenti il meccanismo fa sempre la stessa cosa).

Requisiti irrinunciabili:

1. **Partecipazione Volontaria:** Non devo pagare più di quanto voglio (ossia $P_i(v) \leq v_i$)
2. **Nessuno Escluso a Priori**: Posso sempre ricevere il servizio dicendo che sono "molto interessato" (ossia v_i "grande abbastanza" dove "grande abbastanza" dipende dalle valutazioni degli altri).

Nel caso dell'Asta di 2ndo Prezzo, "grande abbastanza" significa "maggiore delle altre valutazioni". Per l'Asta con Prezzo di Riserva significa "maggiore delle altre valutazioni e del prezzo di riserva". Nel multicast su albero, se facciamo crescere la valutazione di un nodo (agente) ad un certo punto la soluzione con benessere sociale ottimo deve includere questo agente.

4.1 Primo tentativo: prezzi identici

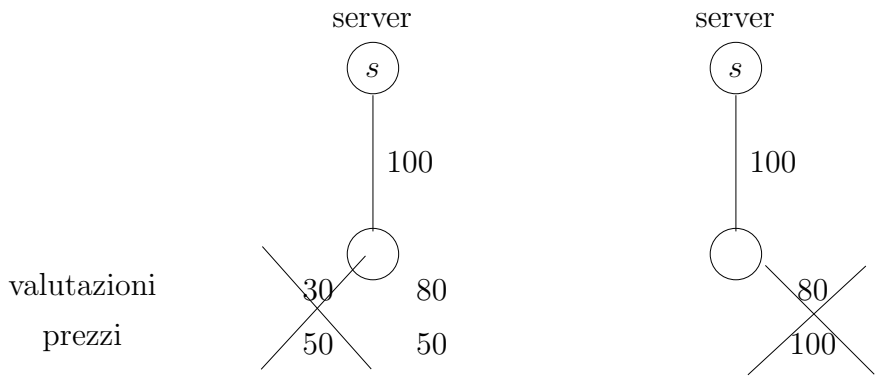
Usiamo prezzi uguali per tutti (come nel primo meccanismo scemo) ma cerchiamo di soddisfare la Partecipazione Volontaria:

1. Prima di dare il servizio, chiediamo se il prezzo va bene

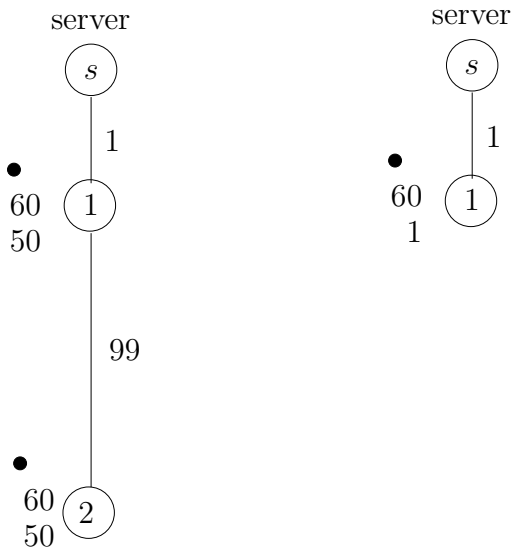
[2]Nel secondo meccanismo usiamo il fatto che se nessuno riceve il costo è zero, ossia $C(\emptyset) = 0$

2. Chi rifiuta l'offerta viene escluso e si ricalcola una nuova offerta per chi è rimasto

Vediamo un esempio:



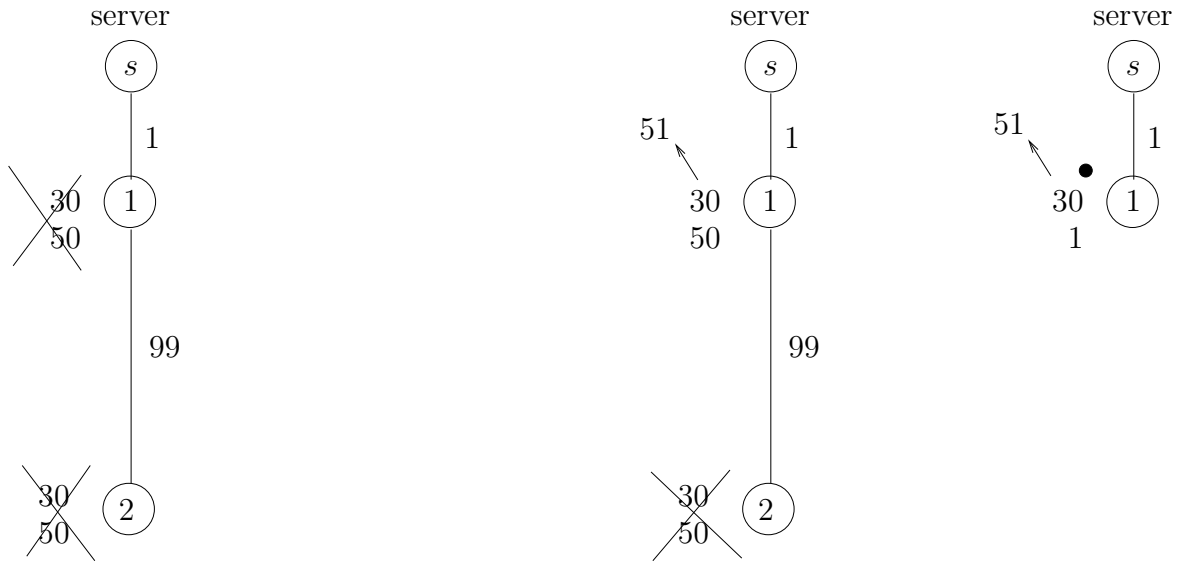
Per il primo agente non avrebbe senso mentire (cercando di rimanere in gioco al primo round) in quanto dovrebbe pagare più della sua vera valutazione. Stessa cosa per l'altro se dichiarasse, ad esempio, 101. Passiamo al muticast su alberi:



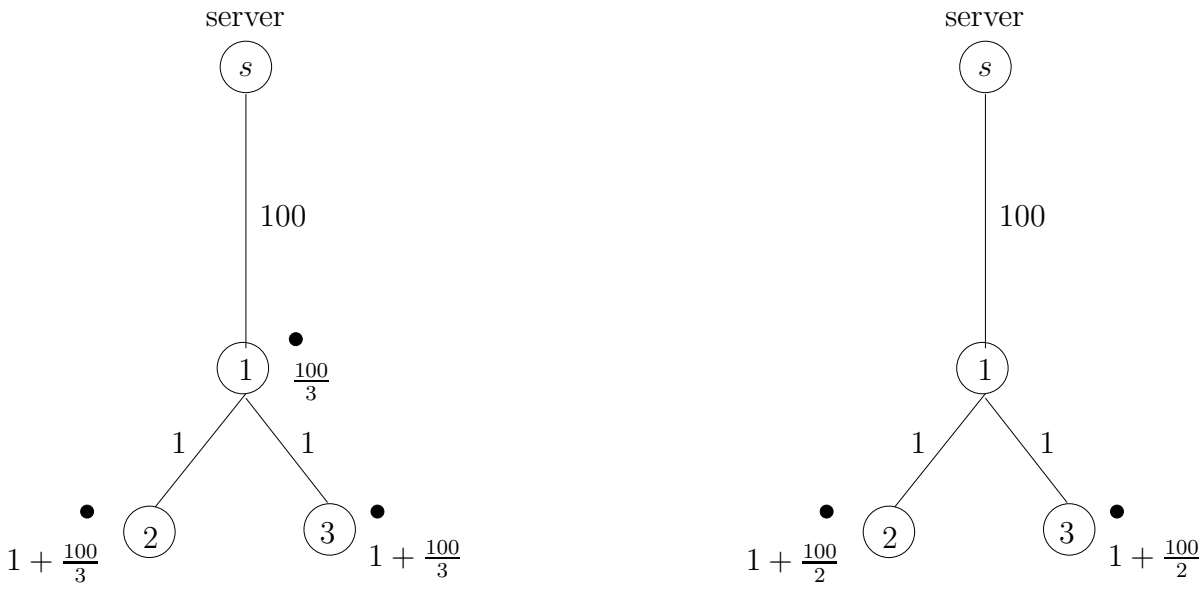
Ora osserviamo che i pagamenti per l'agente 1 nel caso di sinistra sembrano "ingiusti" nel senso che se si trovasse da solo pagherebbe meno (caso a destra).
A questo punto Annibale e Filomena fanno due considerazioni chiave:

> *Annibale*: I pagamenti dovrebbero dipendere dai due archi
>
> *Filomena*: Il nodo "1" potrebbe cacciare fuori "2" e guadagnarci

Partiamo dall'ultima cosa. Benchè l'agente 1 non può fare nulla sull'agente 2, qualcosa di molto simile a quello che dice Filomena in effetti succede:



per queste valutazioni l'agente 2 esce e quindi (mentendo) è meglio per l'agente 1 rimanere dentro al gioco (facendo finta di accettare 50 all'inizio alla fine paga solo 1).
L'osservazione di Annibale ci porta a definire i pagamenti come "ognuno paga il proprio arco": l'agente 1 paga 1 e l'agente 2 paga 99 (entrambi serviti). Applicando questa idea ad un altro esempio ci accorgiamo che a volte un arco è usato da più nodi e potrebbe essere "giusto" far pagare l'arco a tutti quelli che lo utilizzano:



Notiamo una proprietà interessante:

Intuizione:

1. I pagamenti "giusti" dovrebbero essere quelli per cui se un agente esce il prezzo di chi rimane non diminuisce
2. Se abbiamo dei pagamenti "giusti" otteniamo un meccanismo compatibile agli incentivi

4.2 Meccanismi budget-balance basati su pagamenti "giusti"

Finora per ottenere un meccanismo abbiamo seguito questa strada:

Per ottenere Budget-Balance seguiamo il percorso inverso:

Partiamo proprio da quello che vogliamo (Budget-Balance):

$$\sum_{i \in S} P_i(S) = C(S)$$

Da questi pagamenti otteniamo un meccanismo budget-balance che utilizza questi pagamenti per decidere chi viene servito (l'algoritmo):

Meccanismo M_P:

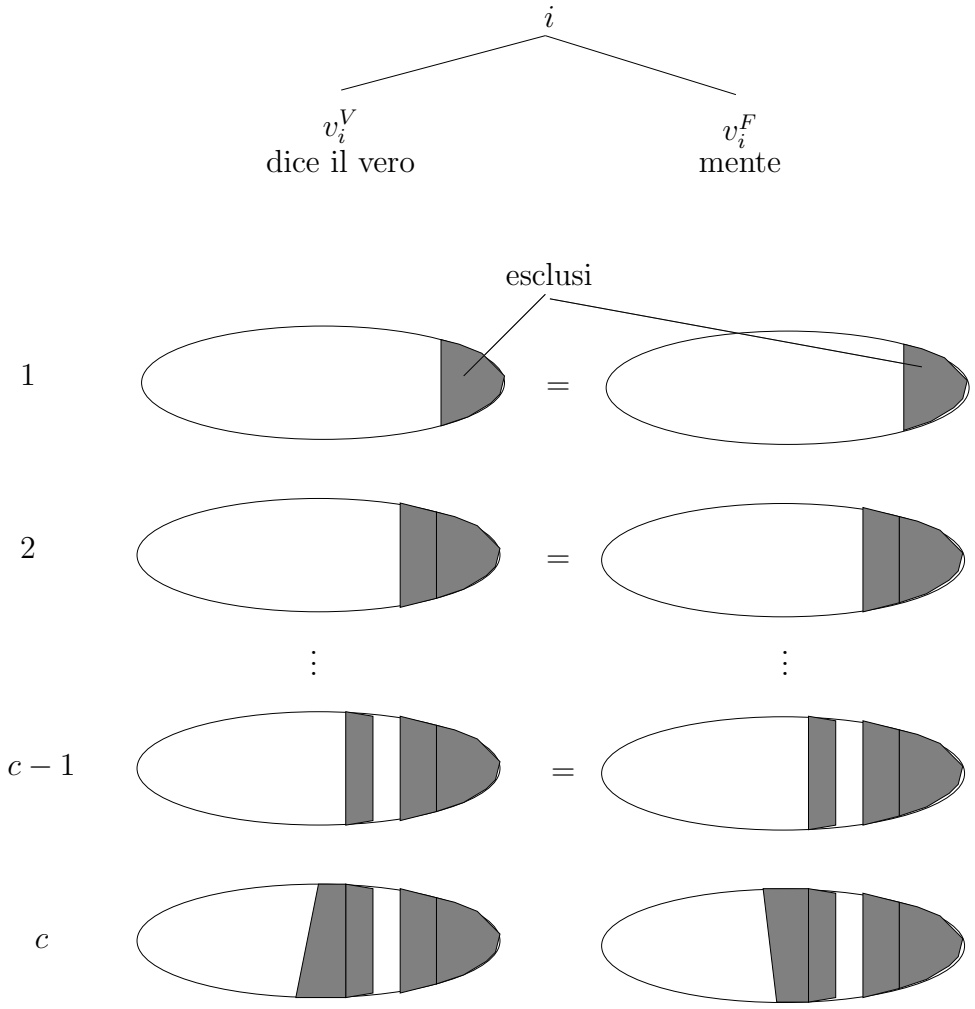
1. Tenta di servire tutti
   $S \longleftarrow U$
2. Ripeti questo passo finchè c'è almeno un agente che rifiuta il prezzo offerto (ossia $v_i < P_i(S)$ per qualche $i \in S$):
   (a) Escludi tutti quelli che rifiutano il prezzo offerto
   $S \longleftarrow S \setminus E$ dove $E := \{j |\ v_j < P_j(S)\}$
3. Ricevono il servizio tutti e soli gli agenti rimasti in S (finale) e pagano $P_i(S)$

Per come abbiamo definito M_P riusciamo anche a soddisfare i due Requisiti Irrinunciabili (vedi pagina 8). Ecco cosa serve per ottenere anche la compatibilità agli incentivi:

Definizione (pagamenti monotoni): I pagamenti P sono monotoni se il prezzo di un agente non diminuisce quando altri agenti vengono esclusi. Ossia

$$\forall S, \forall S' \subset S, \forall i \in S' \quad P_i(S') \geq P_i(S)$$

Teorema: Se i pagamenti P sono monotoni allora il meccanismo M_P è compatibile agli incentivi.
Dimostrazione: Consideriamo un agente i e, fisate le valutazioni v_{-i} degli altri, consideriamo cosa fa il meccanismo quando i dice il vero (valutazione v_i^V) e quando i dice il falso (valutazione v_i^F).
Se l'insieme finale S degli agenti serviti è lo stesso nei due casi, allora anche l'utile di i non cambia (il pagamento dipende solo da S) e quindi mentire non conviene.
Consideriamo il caso in cui l'insieme degli agenti serviti (passo 3 del meccanismo M_P) è diverso nei due casi. Confrontiamo ogni ripetizione del passo 2 di M_P nei due casi:



Ogni ripetizione r del passo 2 il meccanismo prende l'insieme S_{r-1} della ripetizione precedente ed esclude un insieme E_r degli agenti che non accettano il prezzo corrente (quello che otteniamo è un nuovo insieme S_r). Usiamo "V" e "F" per confrontare l'iterazione nel caso vero con il caso falso. All'inizio si parte con i due insiemi uguali (ossia $S_0^V = S_0^F = U$ per il passo 1) ma ad un certo passo c deve succedere che l'insieme degli esclusi è diverso nelle due esecuzioni ($E_c^V \neq E_c^F$). Prendendo come c proprio il la prima iterazione in cui questo succede possiamo dire la cosa seguente. L'insieme S_{c-1}^V e l'insieme S_{c-1}^V sono ancora uguali e quindi ad ogni agente viene offerto lo stesso prezzo nelle due esecuzioni ($P_j(S_{c-1}^F)$ e $P_j(S_{c-1}^V)$ sono ovviamente uguali). La "risposta" degli agenti a questa offerta definisce l'insieme degli esclusi E_c^V ed E_c^F. Abbiamo detto che questi sono diversi ($E_c^V \neq E_c^F$) e questo può succedere solo nel caso in cui l'agente i ha accettato l'offerta in *uno solo* dei due casi: tutti gli altri agenti fanno la stessa cosa perchè abbiamo fissato le loro valutazioni a v_{-i} per entrambe le esecuzioni. Facciamo i due casi possibili:

i **accetta in "vero" e rifiuta in "falso":** Una caratteristica importante del meccanismo è che il prezzo finale (se si viene serviti) non supera mai la valutazione. Nel caso "vero" quindi l'utile è almeno 0 (anche se i rifiutasse un'offerta ad un passo successivo a c avrebbe utile 0). Avendo rifiutato nel caso "falso" l'utile è proprio 0 e quindi mentire non conviene.

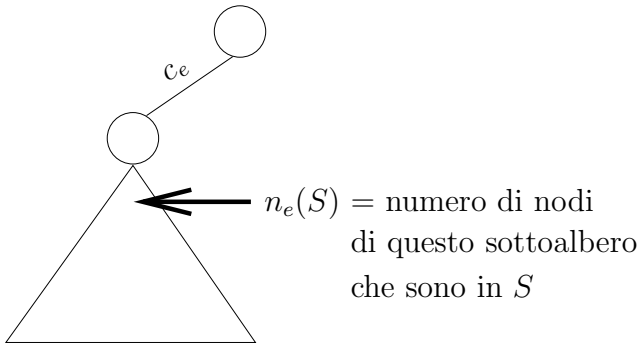
i **accetta in "falso" e rifiuta in "vero":** L'utile nel caso "vero" è 0. Inoltre il fatto che i abbia rifiutato il prezzo al passo c significa che già quel prezzo era troppo alto ($P_i(S_{c-1}^V) >$

v_i^V). Ricordiamoci che nel caso "falso" all'iterazione c viene offerto questo stesso prezzo (perchè $S_{c-1}^V = S_{c-1}^F$) ed ora i accetta questo prezzo. Per la *monotonia* dei prezzi P tutte le offerte successive che il meccanismo fa ad i sono uguali o più alte di $P_i(S_{c-1}^V)$. Se i rifiuta un'offerta successiva l'utile è di nuovo 0 mentre se le accetta tutte è addirittura negativo (accetta e paga un prezzo maggiore della sua vera valutazione). In entrambi i casi mentire non conviene.

Fine della dimostrazione!

4.3 Esempio: multicast su albero

Torniamo al multicast su albero e facciamo vedere che esistono dei pagamenti monotoni (così otteniamo direttamente un meccanismo compatibile agli incentivi e budget-blance).
Ogni arco deve essere pagato tra tutti quelli che lo utilizzano (vedi esempio a pagina 10). Chiamiamo $n_e(S)$ il numero di nodi che utilizzano l'arco e per connettersi al server:



Ogni nodo che utilizza quest'arco deve pagare una frazione del suo costo: $c_e/n_e(S)$. Se un nodo i utilizza gli archi

$$e_1, \ldots, e_k$$

per arrivare al server, allora deve pagare

$$P_i(S) := c_{e_1}/n_{e_1}(S) + \cdots + c_{e_k}/n_{e_k}(S)$$

Per vedere che questi pagamenti sono monotoni basta osservare che se prendiamo $S' \subset S$, il numero di nodi che usa un certo arco non aumenta (ossia $n_e(S') \leq n_e(S)$) e il nodo i deve sempre passare per i k archi elencati sopra perchè la rete è un albero. Quindi $P_i(S') \geq P_i(S)$.
Abbiamo anche budget-balance perchè non abbiamo fatto altro che distribuire (arco per arco) il costo tra gli agenti.

Abbiamo ottenuto un meccanismo M_P per il muticast su alberi che è compatibile agli incentivi e budget-balance.

4.4 Esempio: scheduling

Gli utenti hanno dei task da eseguire e il provider possiede delle macchine su cui si possono eseguire al più 2 task. Il costo per il provider (numero di macchine impegnate) è $C(S) = \left\lceil \frac{|S|}{2} \right\rceil$.
Se dividiamo il costo non otteniamo una cosa monotona:

costo $C(\|S\|)$			
$C(3) = 2$	$2/3$	$2/3$	$2/3$
$C(2) = 1$	$1/2$	$1/2$	
$C(1) = 1$	1		

Ma possiamo facilmente ottenere una versione approssimata di budget-balance:

costo $C(\|S\|)$			
$C(3) = 2$	$2/3$	$2/3$	$2/3$
$C(2) = 1$	1	1	
$C(1) = 1$	1		

Abbiamo cambiato solo la seconda riga (facendo finta che $C(2) = 2$). Così garantiamo $C(S) \leq \sum_i P_i(S) \leq 2C(S)$.