Compromiso entre Pares e ISPs: Un Modelo de Optimización Multiobjetivo

Darío Padula
Director de Tesis: Dr. Pablo Rodriguez-Bocca
Director Académico: Dr. Franco Robledo Amoza
Instituto de Investigación: LPE-IMERL
Universidad de la República
Facultad de Ingeniería
Montevideo,Uruguay

August 31, 2010

1. **Introducción**
   - Redes P2P
   - P4P
2. **Modelo Matemático P4P**
3. **Distintos problemas**
4. **Algoritmos**
   - Generalidades de los Algoritmos para P1
5. **Resultados y Conclusiones**
   - Resultados para P1
   - Emulaciones en GoalBit
   - Algoritmo GRASP vs. cota

Outline

1. **Introducción**
   - Redes P2P
   - P4P
2. **Modelo Matemático P4P**
3. **Distintos problemas**
4. **Algoritmos**
   - Generalidades de los Algoritmos para P1
5. **Resultados y Conclusiones**
   - Resultados para P1
   - Emulaciones en GoalBit
   - Algoritmo GRASP vs. cota

Redes de pares P2P I

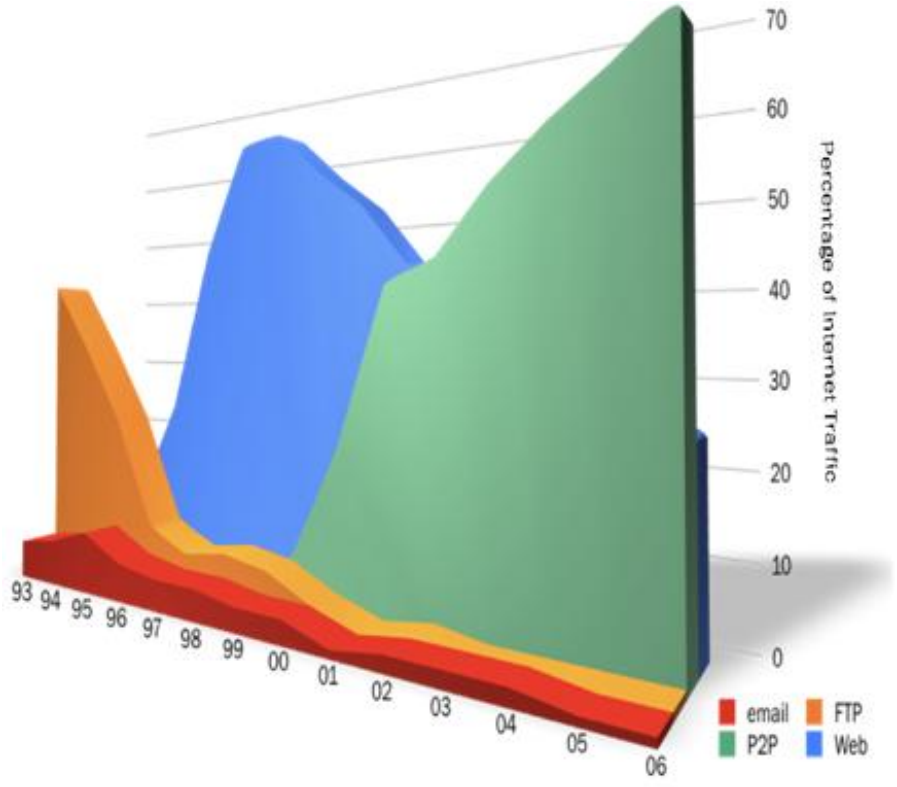
Se caracterizan por:

- Cooperación de pares
- Compartir recursos
- Libre participación
- Múltiples aplicaciones
- Gran consumo de ancho de banda

Surgen por la necesidad de compartir:

- Videos
- Música
- Archivos
- Otras cosas

Redes de pares P2P I



Fuente:P4PWG

Problemas:

- Tráfico excesivo e ineficiente
- Uso indiscriminado de enlaces costosos
- Comunicación congestionada entre pares lejanos

P4P I

Metodología P4P
(*Participación Proactiva del Proveedor en redes P2P*):

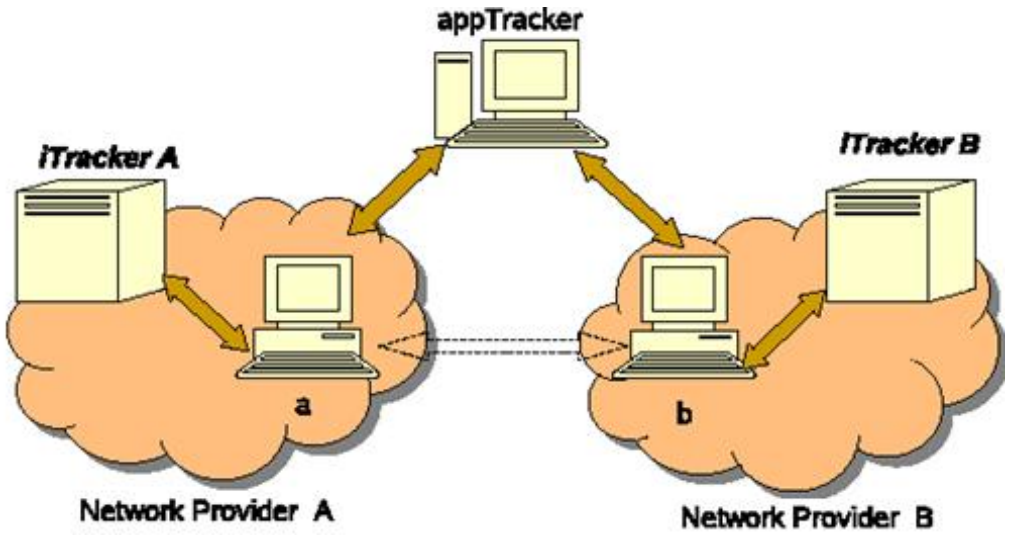
- Surge por el incremento del tráfico debido a las aplicaciones P2P.
- Existe un grupo llamado **P4P Working Group** que estudia su implementación.
- El objetivo de la metodología es lograr mediante la comunicación explícita entre aplicaciones P2P e ISPs:
  - una mejor performance en el tráfico de contenidos P2P, y
  - un uso más eficiente de los recursos de la red.

P4P I

Características de diseño *P4P*:

- **Mejor performance para las redes P2P** (usando información más exacta del estado de la red)
- **Uso eficiente de los recursos de la red** (permitiendo una comunicación explícita entre las aplicaciones P2P y los ISPs)
- **Escalabilidad** (soporta un número grande de usuarios y de redes P2P en un sentido dinámico)
- **Robustez** (fallas en las componentes P4P pueden bajar la eficiencia, pero no cae el sistema)
- **Garantiza privacidad**
- **Compatibilidad** (P4P puede manejar un amplio rango de aplicaciones P2P con características variadas)

Componentes de P4P y funcionamiento I



1. Los pares consultan al iTracker por la topología y la política
2. Luego le envían esta información al appTracker
3. El appTracker selecciona los pares, considerando la información del iTracker y los requerimientos de la aplicación

Objetivos contrapuestos I

1. Los pares quieren maximizar el tráfico sin preocuparse por los recursos disponibles de la red.
2. Los ISP quieren minimizar el tráfico por los enlaces más costosos.

Topología I

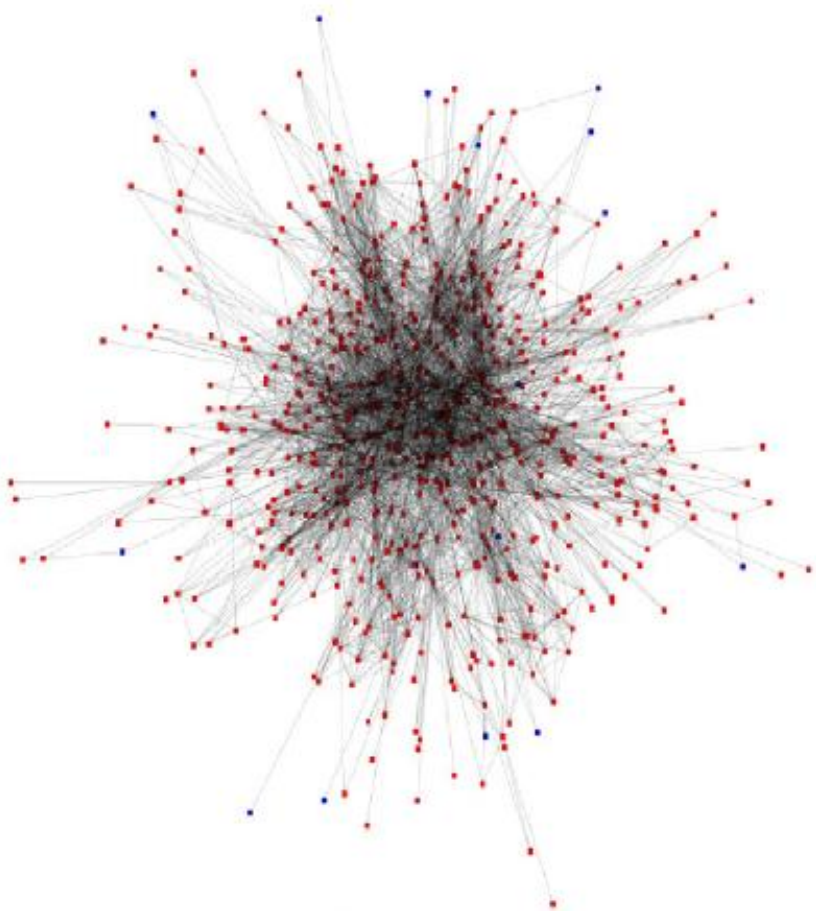


Figure: Sin P4P

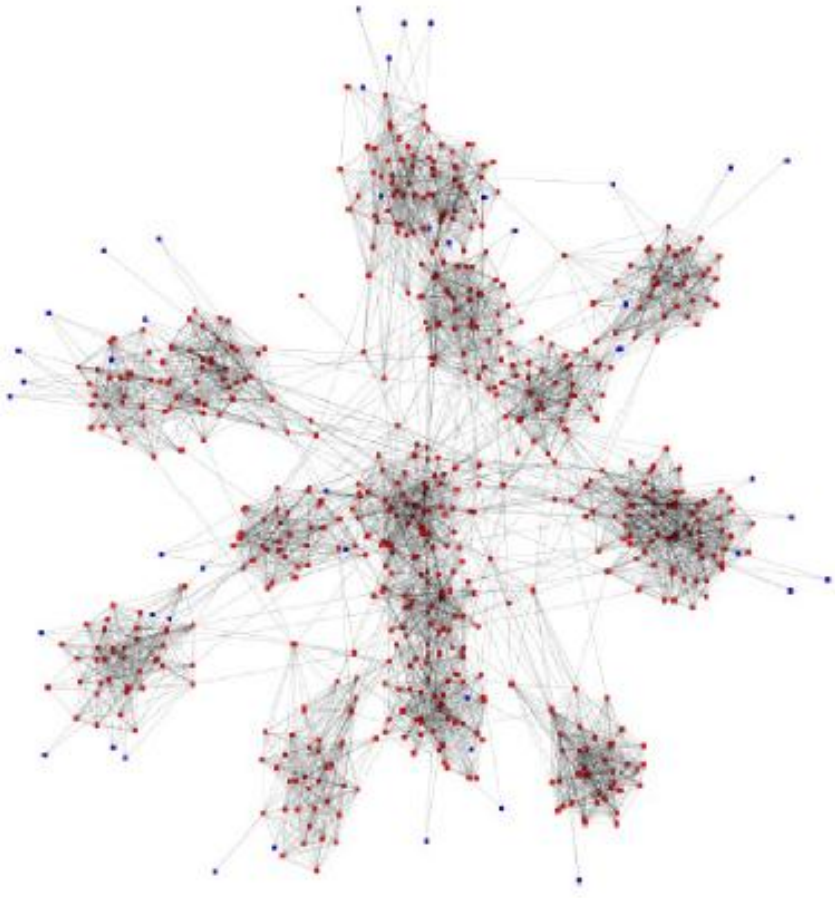


Figure: Con P4P

Outline

1. Introducción
   - Redes P2P
   - P4P
2. **Modelo Matemático P4P**
3. Distintos problemas
4. Algoritmos
   - Generalidades de los Algoritmos para P1
5. Resultados y Conclusiones
   - Resultados para P1
   - Emulaciones en GoalBit
   - Algoritmo GRASP vs. cota

Notación I

- Llamaremos *PID* a un conjunto de pares. Cada *PID* va a ser un vértice de un grafo. **En particular nosotros asumimos que cada *PID* es el conjunto de pares dentro de un mismo país**.
- **Las aristas son enlaces internacionales (conectan países)**.

Notación I

Parámetros:

- PID_i es el i-esimo grupo de pares (países).
- b_e es el tráfico de fondo de la arista e (**Background traffic**).
- c_e es la capacidad de la arista e.
- $I_e(i,j)$ indica si la arista e está en la ruta del PID_i al PID_j.
- u_i^k y d_i^k capacidades de subida y bajada del PID_i para el contenido k.

Variable de decisión:

- t_{ij}^k tráfico neto del PID_i al PID_j para el contenido k.

Objetivos I

Objetivo de los pares

$$\begin{cases} max \sum_i \sum_{j \neq i} t_{ij}^k \\ s.a. \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i \\ \quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i \\ \quad t_{ij}^k \geq 0 \quad \forall i \neq j \end{cases}$$

Objetivo de los *ISP*

$$\begin{cases} minmax_{e \in E} \{ b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times l_e(i,j)}{c_e} \} \\ s.a. \quad t_{ij} \geq 0 \quad \forall i \neq j \end{cases}$$

Optimización multiobjetivo P4P I

De esta forma se combinan ambos deseos en un mismo problema multiobjetivo:

$$\begin{cases} minmax_{e \in E}\{b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times l_e(i,j)}{c_e}\} \\ s.a.\ \max \sum_i \sum_{j \neq i} t_{ij}^k \quad \forall k \\ \quad \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i \\ \quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i \\ \quad t_{ij} \geq 0 \quad \forall i \neq j \\ \quad \sum_k \sum_i \sum_{j \neq i} t_{ij}^k \times l_e(i,j) \leq v_e \quad \forall e \end{cases}$$

Se agrega la restricción $\forall e$ (enlace internacional) donde $v_e = c_e(1 - b_e)$ es la **capacidad virtual**.

Outline

1 Introducción
- Redes P2P
- P4P

2 Modelo Matemático P4P

3 Distintos problemas

4 Algoritmos
- Generalidades de los Algoritmos para P1

5 Resultados y Conclusiones
- Resultados para P1
- Emulaciones en GoalBit
- Algoritmo GRASP vs. cota

Problemas a abordar I

Definimos 6 problemas:

1. **Problema General o PG:**
   - Múltiples contenidos.
   - Conocimiento completo de la red.
2. **Problema 1 o P1:**
   - Sólo un contenido.
   - Conocimiento completo de la red.
3. **Variante 1 del P1 ($VP1_\rho$):**
   - Utilización máxima fija ρ_{max}
   - Conocimiento completo de la red.
4. **Variante 2 del P1 ($VP1$)**
   - Sólo ruteos por caminos de largo 1.
   - Conocimiento completo de la red.

Problemas a abordar II

5. **Problema 2 o P2:**
   - Sólo un contenido.
   - Se conoce el tráfico de fondo sólo de los enlaces de un *PID*.
6. **Problema 3 o P3:**
   - Sólo un contenido.
   - Se conoce el tráfico de fondo sólo de los enlaces de un *PID*.
   - Se conoce las capacidades sólo de los enlaces de un *PID*.

Outline

1 Introducción
- Redes P2P
- P4P

2 Modelo Matemático P4P

3 Distintos problemas

4 Algoritmos
- Generalidades de los Algoritmos para P1

5 Resultados y Conclusiones
- Resultados para P1
- Emulaciones en GoalBit
- Algoritmo GRASP vs. cota

Algoritmos

- **Algoritmo "*MínimoEnlace*" para resolver el Problema 1.**
- **Algoritmo basado en GRASP para resolver el Problema General, el cual consta de dos fases:**
  1. **Algoritmo "RandomList"** para la Fase 1 (construcción)
  2. **Algoritmo "BL"** para la Fase 2 (Búsqueda Local)

Para el Problema 1 se estudiaron también otros algoritmos...

Algoritmo "*MínimoEnlace*" para resolver el P1 I

Generalidades del Algoritmo "*MínimoEnlace*":

- Se construye una red auxiliar para transformar el P1 en un problema de flujo máximo - corte mínimo.
- Mediante el Algoritmo de Ford-Fulkerson se calcula el flujo máximo (F_{max}) que pueden introducir los *PIDs* en la red.
- Aplicando el Teorema de Bolzano encontramos la utilización mínima de los enlaces (α_{opt})

Red Auxiliar I

Ejemplo de la construcción de $R = (V, E, C)$:

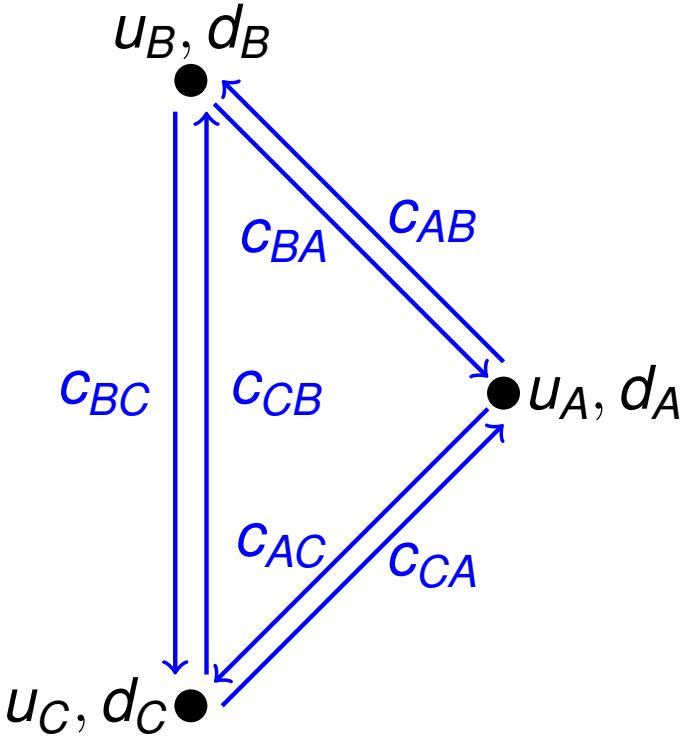


Figure: Red original

Red Auxiliar I

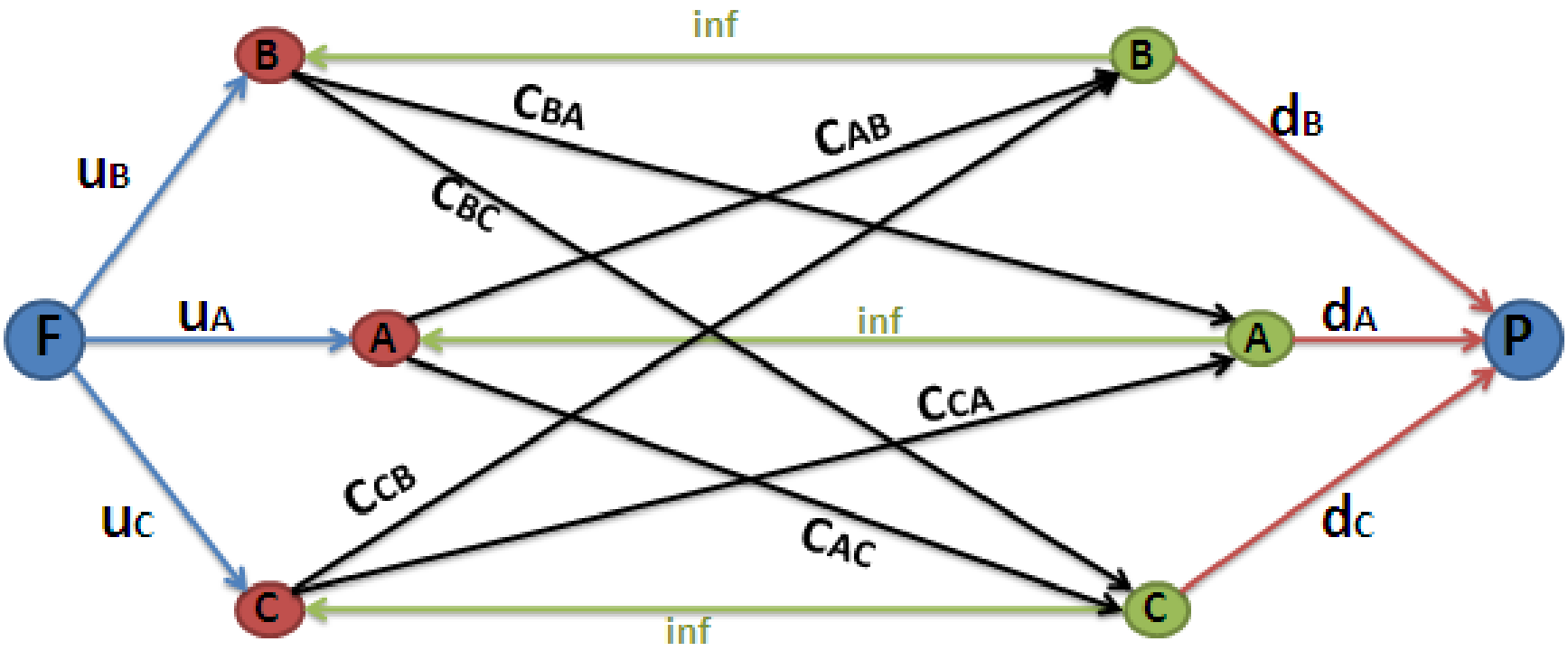


Figure: Adaptación para flujo máximo (Red Auxiliar)

Red Auxiliar I

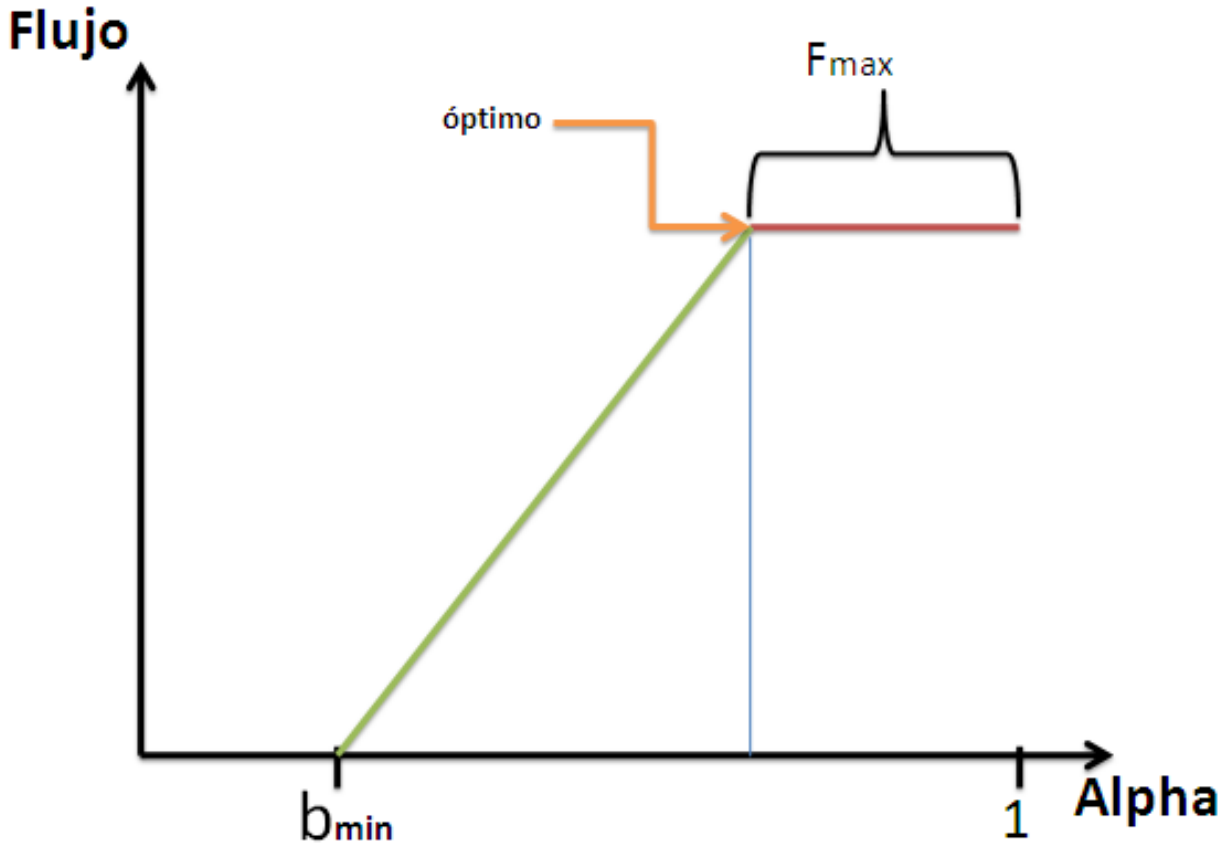
Para transformar el Problema 1 como un problema de Flujo máximo - Corte mínimo se construye la red auxiliar $R = (V, E, C)$**:**

1. Se hace una duplicación de los *PIDs* (nodos) de la red (Transmisores y Receptores).
2. Se agregan dos nodos auxiliares, *Fuente* y *Pozo* que llamaremos F y P respectivamente.
3. Se determinan los enlaces y sus capacidades de tal forma que:
   - se verifiquen las restricciónes del Problema 1, y
   - exista una biyección entre la red original y la auxiliar.

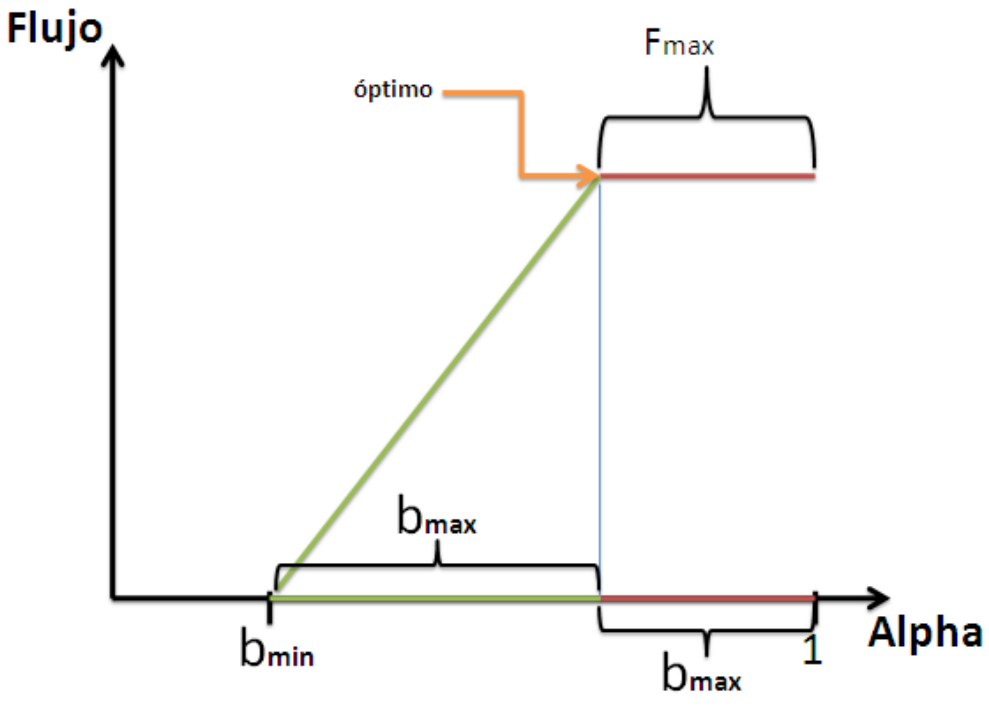
Pseudo-Código: "*MínimoEnlace*" I

Entradas: U, D, C, B, ε, F_{max}
Inicialización: $h = 0$, $L_{inf} = b_{min}$, $L_{max} = 1$
while $(\frac{1-b_{min}}{2^h} > \varepsilon)$ **do**
    $\alpha_h = \frac{L_{min}+L_{max}}{2}$
    $F_i \leftarrow$ F-F con $c'_{i,j} = max\{c_{i,j}(\alpha_h - b_{i,j}), 0\}\ \forall i \neq j$
    if $F_i = F_{max}$ **then**
        $L_{max} = \alpha_h$
    else if $F_i < F_{max}$ **then**
        $L_{min} = \alpha_h$
    end if
    $h = h + 1$
end while
return $\alpha_{opt} \in [b_{min}, 1]$, *Caminos*, *magnitudes*

Pasos del algoritmo I



Pasos del algoritmo I



El porcentaje de saturación máximo en la red es $\rho_{max} = max\{\alpha_{opt}, b_{max}\}$, donde $b_{max} = max_e\{b_e\}$.

Algoritmo GRASP para el PG I

Generalidades del Algoritmo:
Se aplica un Algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*) el cual consta de dos fases:

- **Fase 1 (Construcción):** sortea un contenido k con probabilidad p_k e introduce tanto flujo como sea posible aplicando el Algoritmo "*MínimoEnlace*".
- **Fase 2 (Búsqueda Local):** reduce la utilización del enlace más congestionado (MLU).

Algoritmo GRASP para el PG II

En la Fase de construcción cuando se sortea el $h-$ ésimo contenido, se determina el ruteo óptimo para ese contenido dado que ya fueron introducidos los $h-1$ contenidos anteriores.
El orden en que se introducen los contenidos influye en la calidad de la solución (sorteo aleatorio priorizando algún orden).

Fase 1: Pseudo-Código "*RandomList*" I

1: **Entradas:** U^k, D^k, C, B, ε, p
2: **Inicialización:** $P \leftarrow \emptyset$, $m \leftarrow \emptyset$, $H \leftarrow ceros_{n \times n}$
3: **while** $(\sum_i p_i > 0)$ **AND** $(C > 0)$ **do**
4: $k \leftarrow SorteoAleatorio(p)$
5: $(Cam, mag) \leftarrow MinimoEnlace(U^k, D^k, C, B + H, \epsilon)$
6: $H \leftarrow AgregaFlujo(Cam, mag, H)$
7: $P \leftarrow P \cup Cam$; $m \leftarrow m \cup mag$
8: $p_k \leftarrow 0$; Se normaliza p
9: **end while**
10: **return** P, m

Fase 2: Pseudo-Código "*BL*" I

1: **Entradas:** U^k, D^k, C, B, ε, P, m, $iter$
2: **Inicialización:** $e_{sol} \leftarrow 1$, $P_{sol} \leftarrow P$, $m_{sol} \leftarrow m$, $j \leftarrow 1$
3: **while** (Existan mejoras) **AND** ($j < iter$) **do**
4: $e_{sat} \leftarrow MaximoEnlace(P_{sol}, m_{sol})$; $C' \leftarrow C_{e_{sat}} = 0$
5: **for** Contenidos $i \in e_{sat}$ **do**
6: $(\tilde{F}^i_{max}, Cam, mag)$ =$MinEnl(U^i, D^i, C', B + (H - H^i), \epsilon)$
7: $e_i \leftarrow MaximoEnlace(Cam, mag)$
8: **if** ($e_i < e_{sat}$) **AND** ($e_i < e_{sol}$) **AND** ($\tilde{F}^i_{max} = F^i_{max}$) **then**
9: **Actualizar:** P_{sol}; m_{sol}; e_{sol}
10: **end if**
11: **end for**
12: **end while**
13: **return** P_{sol}, m_{sol}

Goloso Simple I

Generalidades del Algoritmo Goloso Simple:

- Elige un PID_i al azar para subir flujo a la red.
- Elige un PID_j con $j \neq i$ para descargar flujo.
- El PID_i sube todo el flujo que puede al PID_j.

Ventajas:

- La solución que se obtiene siempre es factible.
- Es muy fácil de implementar.

Desventajas:

- No garantiza alcanzar el objetivo de los pares (sólo utiliza caminos de largo 1).
- No realiza un reparto equitativo por los distintos enlaces.
- Tiende a saturar siempre algún enlace.

GOTEO I

Generalidades del Algoritmo de GOTEO:

- Elige un PID_i al azar para subir flujo a la red.
- El PID_i sube todo el flujo que puede a sus *PID* vecinos utilizando *waterfilling*.

Ventajas:

- La solución que se obtiene siempre es factible.
- La transferencia de flujo por los enlaces se realiza de forma equitativa (*waterfilling*).
- Es muy fácil de implementar.

Desventajas:

- No garantiza alcanzar el objetivo de los pares (sólo utiliza caminos de largo 1).

Algoritmo "*MínimoEnlace*" I

Generalidades del Algoritmo "*MínimoEnlace*":
Ventajas:

- La solución que se obtiene siempre es factible.
- Alcanza soluciónes tan próximas como queramos a las óptimas (*FPTAS*).
- La transferencia de flujo por los enlaces se realiza de forma equitativa nivelando el porcentaje de utilización.
- Es muy fácil de implementar.

Desventajas:

- Se pueden obtener soluciones con tráficos redundantes (ciclos) pero existen métodos para solucionar este problema.

Outline

1. Introducción
   - Redes P2P
   - P4P
2. Modelo Matemático P4P
3. Distintos problemas
4. Algoritmos
   - Generalidades de los Algoritmos para P1
5. **Resultados y Conclusiones**
   - Resultados para P1
   - Emulaciones en GoalBit
   - Algoritmo GRASP vs. cota

Distintos resultados I

Se obtuvieron tres tipos de resultados distintos:

1. Comparaciones de distintos algoritmos para resolver el Problema 1:
   - Goloso Simple
   - GOTEO
   - "*MínimoEnlace*"
2. Emulaciones en la plataforma real GoalBit (GoalBit-emu).
3. Comparaciones de Algoritmo GRASP con una cota conocida para las distintas instancias (múltiples contenidos PG).

Comparación de algoritmos I

Instancia	GS	GOTEO	FF	Saturación GOTEO	% Inc. FF vs GOTEO	% Inc. FF vs GS
1	13096	13101	13101	0.9950	0.0000	0.0382
2	11442	11447	11447	0.9963	0.0000	0.0437
3	29949	29960	29960	0.9973	0.0000	0.0367
4	31916	32048	32048	0.9909	0.0000	0.4136
5	64423	64409	65183	0.9931	1.2017	1.1797
6	40465	40512	40512	0.9950	0.0000	0.1161
7	66549	66583	66600	0.9970	0.0255	0.0766
8	27600	27625	27625	0.9957	0.0000	0.0906
9	34692	34692	34692	1.0000	0.0000	0.0000
10	101095	101095	101880	1.0000	0.7765	0.7765
11	38804	38804	38804	1.0000	0.0000	0.0000
12	110577	110544	110855	0.9955	0.2813	0.2514

Table: Desempeño de los algoritmos: Goloso simple, de GOTEO y el basado en Ford-Fulkerson ante distintas instancias fijando la utilización alcanzada por el Algoritmo de GOTEO.

Instancias para emular I

Características de las 9 instancias:

- Se extrajeron de distintos streamings de partidos de fútbol de la Liguilla Uruguaya del 2006.
- Se consideraron distintas ventanas de tiempo.
- En cada instancia los rangos de pares que se conectan y desconectan son variados.

Para las emulaciones se comparó el desempeño real del streaming aplicando P4P (*Algoritmo Simplex*) vs. aplicar una selección de pares al azar.

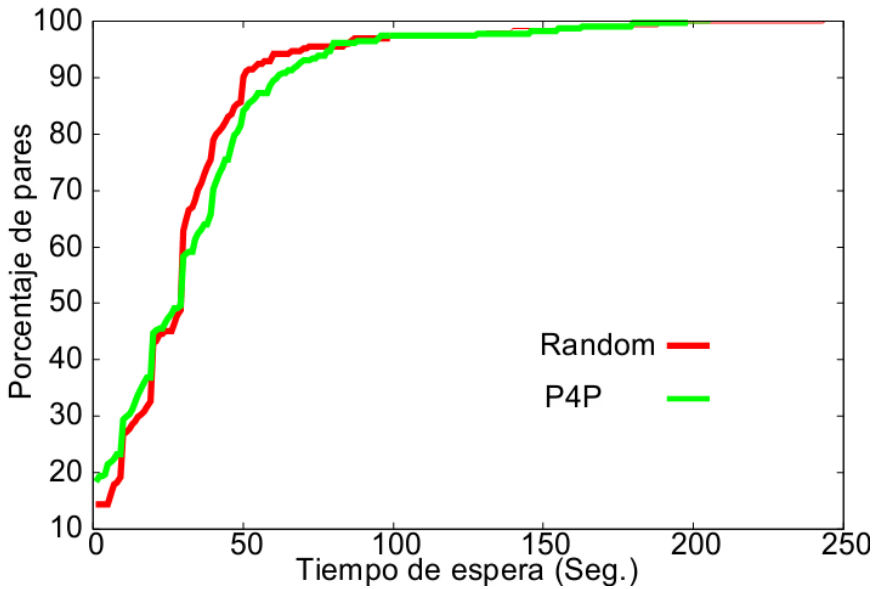
Resultados de las emulaciones en GoalBit I

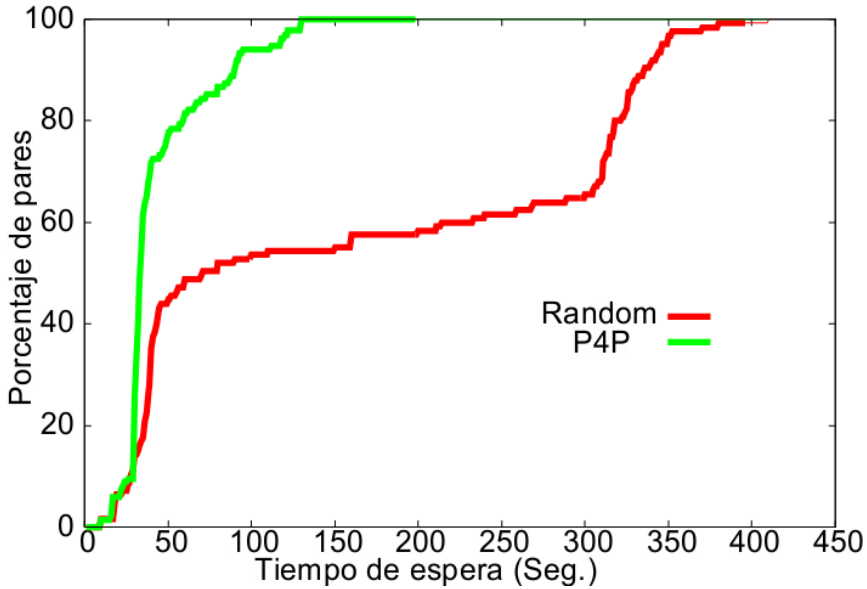
Instancia	Rango pares	Δ_{FS}	Δ_{FE}	Δ_{FT}
1	20-40	13.70	-59.02	2.49
2	20-40	-14.63	54.26	6.29
3	20-40	-13.05	13.35	-2.42
4	20-40	-23.22	49.10	-3.52
5	60-120	-47.57	-74.96	0.00
6	60-120	-43.33	-44.55	-1.86
7	75-80	-45.64	-59.50	-12.17
8	80-200	-38.80	-58.08	19.49
9	100-110	-56.59	-35.63	6.93
Media	70	-29.90	-23.89	1.69
Desvío	40	22.35	49.58	8.79

Table: Resultados de las 9 emulaciones realizadas en la red GoalBit.

Resultados de las emulaciones en GoalBit I

Distribución de los tiempos de espera (seg.) para las instancias 5 y 9.





Cota considerada I

Construcción:

- Se toman todos los contenidos como un único contenido.
- Se aplica el Algoritmo basado en Ford-Fulkerson ("*MínimoEnlace*") para el tráfico del contenido genérico.

Características:

- Es una cota superior para el flujo máximo.
- Si alguna solución del PG obtenida por algún algoritmo alcanza:
  - **la utilización y el flujo** $\Rightarrow$ óptima global.
  - **el flujo PERO NO la utilización** $\Rightarrow$ óptimo *PIDs* y podemos obtener una cota del GAP entre la saturación alcanzada y la óptima.
  - **Nl el flujo Nl la utilización** $\Rightarrow$ sólo podemos calcular una cota superior del GAP entre el flujo obtenido y el óptimo.

Instancias para el PG I

La instancias para evaluar el PG se construyeron de forma aleatoria, siguiendo el siguiente procedimiento:

- $U = rand(K, N) \times N$, donde N es el total de *PIDs*.
- $D = rand(K, N) \times r \times N$, siendo r una constante de diseño.
- $C = rand(K, N) \times s \times K$, donde s regula la holgura de la red.
- Suponemos que no existe tráfico de fondo: $B = \emptyset$

En valor esperado cada *PID* tendrá una magnitud de flujo para subir a la red igual a $K \times \frac{N}{2}$.

Resultados PG vs. cota I

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
15	5	0.9667	0.0676	559.6	11.14	1	0	100
15	10	0.7445	0.0637	1106.6	42.83	1	0	100
30	5	0.8776	0.0774	2239.3	103.90	1	0	100
30	10	0.7696	0.0700	4498.9	197.37	1	0	100

Table: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$, $s = 2$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.8409	0.0709	973.6	54.16	0.995	0.0112	80
20	10	0.8498	0.0906	2000.0	86.83	0.9863	0.0085	0
40	5	0.9297	0.0242	4081.9	106.90	0.9972	0.0054	60
40	10	0.8428	0.0308	7932.6	86.71	0.9948	0.0050	40

Table: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.2$, $s = 1.8$, $iter = 10$ y $veces = 5$

Resultados PG vs. cota I

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.9321	0.0627	1025.9	83.26	1	0	100
20	10	0.9128	0.0847	1965.9	67.28	0.9983	0.0024	60
40	5	0.9850	0.0335	3938.7	74.22	1	0	100
40	10	0.8761	0.0735	8050.2	197.79	0.9984	0.0037	80

Table: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.5$, $s = 1.7$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	1.0000	0.0000	869.6	40.76	0.9982	0.0024	40
20	10	1.0000	0.0000	1699.9	72.49	0.9961	0.0024	0
40	5	1.0000	0.0000	3512.9	75.20	0.9971	0.0019	0
40	10	1.0000	0.0000	7233.1	160.49	0.9972	0.0019	0

Table: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$, $s = 1$, $iter = 10$ y $veces = 5$

Conclusiones I

- Se desarrolló una Algoritmo *FPTAS* para resolver el Problema 1 (un contenido).
- Los resultados de las emulaciónes en GoalBit justifican el uso de P4P, ya que se alcanzó reducir el tráfico de flujo por los enlaces costosos sin perdida de transferencia ni de calidad del servicio.
- Diseñamos un algoritmo para el Problema General (múltiples contenidos) que registra un muy buen desempeño ante las instancias generadas.
- Se definió una cota para el flujo máximo la cual es de gran utilidad para comparar resultados de soluciones del Problema General.

Conclusiones I

Todo indica que si se implementa la metodología P4P utilizando el Algoritmo basado en GRASP, los ISPs obtengan grandes beneficios económicos y que las aplicaciones P2P se desempeñen de forma más eficiente.

Preguntas?

Muchas Gracias!!