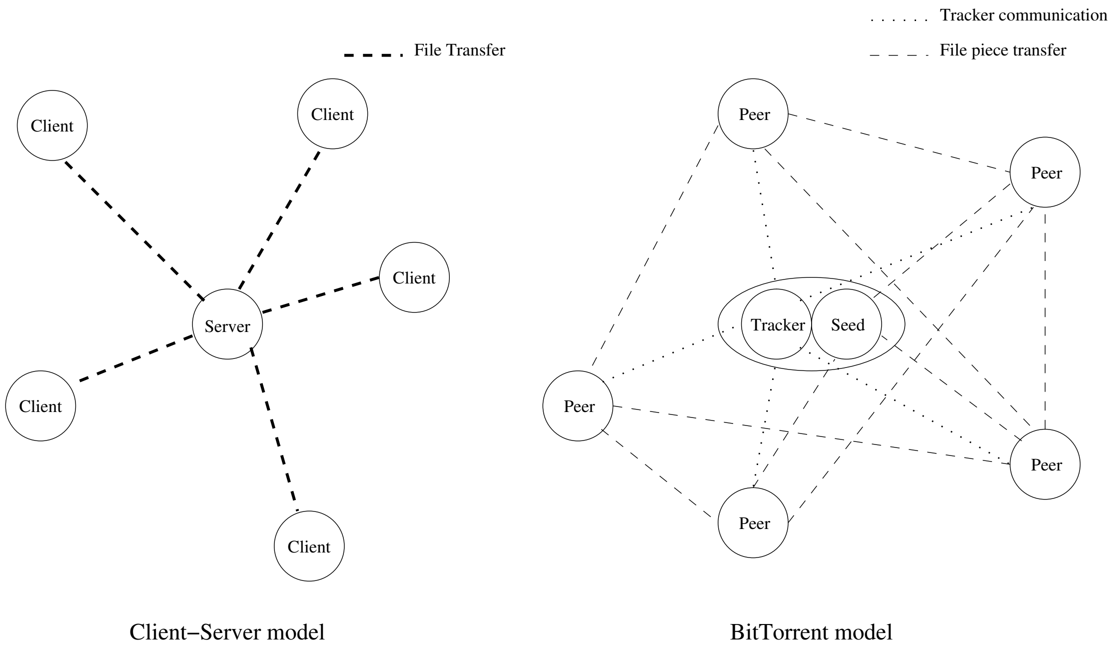
Modelling BitTorrent in PEPA

Adam Duguid

September 6, 2005

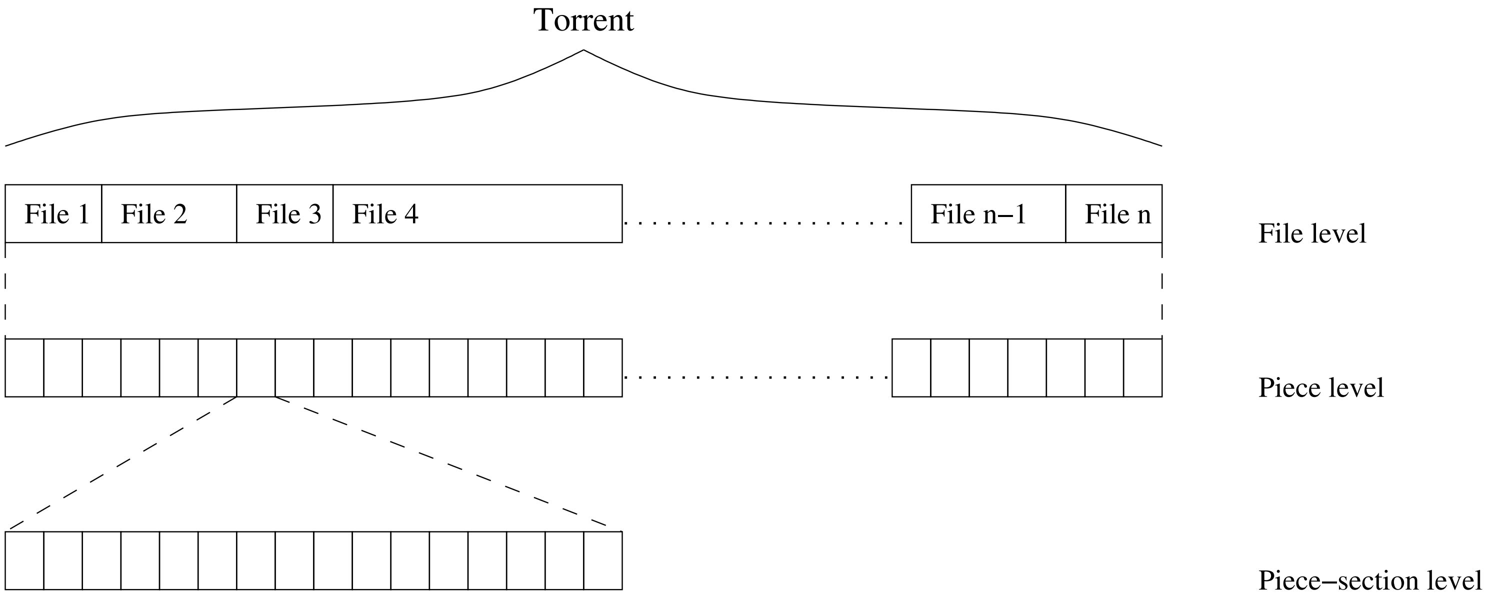
School of informatics

BitTorrent



- All peers connect to tracker for peer discovery.
- Transfer between two peers (direct connections) uses the tit-for-tat strategy.
- Transfers happen concurrently with multiple peers to improve efficiency.

BitTorrent



- File is split into pieces and transferred using rarest-first strategy to help ensure distribution.

PASTA'04 - Modelling BitTorrent: Huge Model

$$
\begin{aligned}
Piece_{i,j} &= (transfer_{1,i}, download).GotPiece_{i,j} + \ldots \\
&+ (transfer_{I,i}, download).GotPiece_{i,j}; \\
GotPiece_{i,j} &= (transfer_{i,1}, upload).GotPiece_{i,j} + \ldots \\
&+ (transfer_{i,J}, upload).GotPiece_{i,j}; \\
Peer_i &= (Piece_{i,1} \mid\mid \ldots \mid\mid Piece_{i,J}) \\
System &= (Peer_i < \mathrm{L} > \ldots < \mathrm{L} > Peer_I < \mathrm{L} > Seed)
\end{aligned}
$$

$$
\begin{aligned}
i &= peer_i \\
I &= \#peers \\
J &= \#pieces \\
j &= piece_j
\end{aligned}
$$

State space size $\approx \left(2^{\#pieces}\right)^{\#peers}$

PASTA'04 - Modelling BitTorrent: Smaller Model

$$
\begin{aligned}
Peer_{i,j} &= (transfer_{1,i}, download).Peer_{i,j+1} + ... \\
&+ (transfer_{n,i}, download).Peer_{i,j+1} \\
&+ (transfer_{i,1}, upload).Peer_{i,j} + ... \\
&+ (transfer_{i,n}, upload).Peer_{i,j};
\end{aligned}
$$

$$
\begin{aligned}
i &= peer_i \\
I &= \#peers \\
J &= \#pieces \\
j &= piece_j
\end{aligned}
$$

$$
System = (Peer_{1,0} < \mathrm{L} > ... < \mathrm{L} > Peer_{I,0} < \mathrm{L} > Peer_{I+1,J})
$$

State space size $\approx \#pieces^{\#peers}$

PEPA and ISBJava

As pieces of the file can be downloaded in any order (Huge Model) we have:

- $\#piece!$ orders in which $\#piece$ can be downloaded
- $2^{\#pieces}$ permutations of obtained pieces *for one peer*
- State space of size $\left(2^{\#pieces}\right)^{\#peers}$ when taking into account number of peers for the actual BitTorrent protocol.
- State space of size $100^{\#peers}$ if simplified to % complete (Smaller Model).

Only one small problem...

PEPA and ISBJava

Only one small problem...

number of atoms in the universe $\approx 10^{78}$-10^{81} compared to $100^{\#peers}$

Constructing the state space in PEPA isn't the way to go.

Instead we shall use the reagent-centric approach as proposed by Calder, Hillston and Gilmore.

Use the ISBJava library to create a reaction based model from reagent-centric PEPA.

Goals of the Project

1. Based on the reagent-centric approach, formulate a process to convert between PEPA and ISBJava.

2. Based on the conversion process, implement as a module for Choreographer.

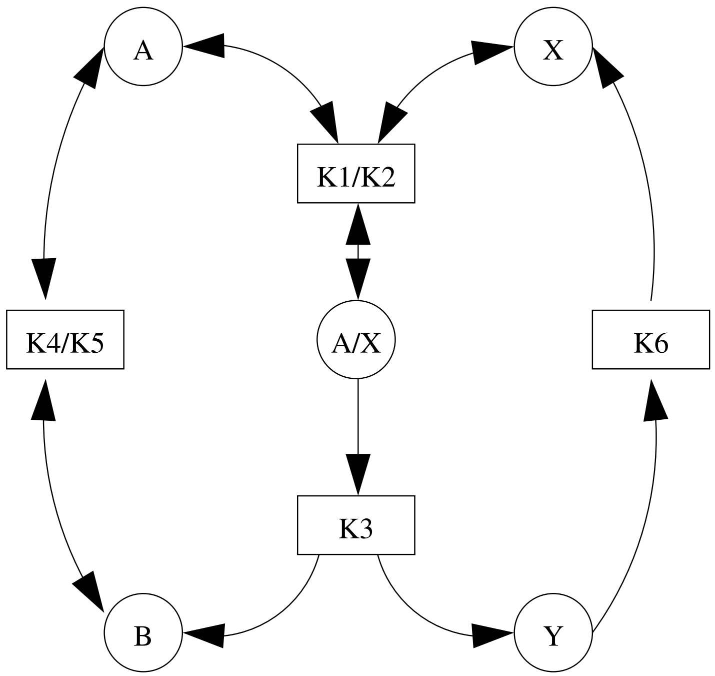
3. Create a BitTorrent model based on the reagent-centric model and analyse.

Conversion Process

1. Identify species and population size declarations from defined components and rates.
2. Zero all population sizes not previously declared.

   At this stage verification of component declarations has occurred guaranteeing declarations have a 'direction'
3. Form reactions from component declarations.
4. Modify rate expressions to conform to rate transformations within step 1.
5. Gather stoichiometric information.
6. Reaction rate expressions altered to include delays linked to population sizes or densities.

Walkthrough example [CGH CMSB'05]



k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react,k1).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

1. Identify species and population size declarations

Rates

Species

A = 1000; X = 500;

AX = 200;

B Y

Reactions

k1 = 1; k2 = 1; k3 = 1;

k4 = 2*k3; k5 = 2*x; k6 = 2*b;

a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L ;

A_L = (k2react,k2).A_H + (k4react,k4).A_H ;

X_H = (k1react,k1).X_L ;

X_L = (k2react,k2).X_H + (k6react,k6).X_H ;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L ;

AX_L = (k1react,k1).AX_H ;

B_H = (k4react,k4).B_L ;

B_L = (k5react,k5)B_H + (k3react,k3).B_H ;

Y_H = (k6react,k6).Y_L ;

Y_L = (k3react,k3).Y_H ;

2. Zero all population sizes not previously declared

Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0 ;

Reactions

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react,k1).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

3. Form reactions from component declarations

Rates
k1 = 1; k2 = 1; k3 = 3;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react , A -> , k1 ;

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react ,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react,k1).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

3. Form reactions from component declarations

Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react , A + X -> , k1 ;

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react ,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

3. Form reactions from component declarations

Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react , A + X -> AX , k1 ;

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react ,k1).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

3. Form reactions from component declarations

```
Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react, A + X -> AX, k1;
k2react, AX -> A + X, k2;
k3react, AX -> B + Y, k3;
k4react, B -> A, k4;
k5react, A -> B, k5;
k6react, Y -> X, k6;
```

```
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;
A_L = (k2react,k2).A_H + (k4react,k4).A_H;
X_H = (k1react,k1).X_L;
X_L = (k2react,k2).X_H + (k6react,k6).X_H;
AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;
AX_L = (k1react,k1).AX_H;
B_H = (k4react,k4).B_L;
B_L = (k5react,k5)B_H + (k3react,k3).B_H;
Y_H = (k6react,k6).Y_L;
Y_L = (k3react,k3).Y_H;
```

4. Modify rate expressions

```
Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*X ; k6 = 2*B ;

Species
A = 1000; X = 500 ;
AX = 200;
B = 0 ; Y = 0;

Reactions
k1react, A + X -> AX, k1;
k2react, AX -> A + X, k2;
k3react, AX -> B + Y, k3;
k4react, B -> A, k4;
k5react, A -> B, k5;
k6react, Y -> X, k6;
```

```
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;
A_L = (k2react,k2).A_H + (k4react,k4).A_H;
X_H = (k1react,k1).X_L;
X_L = (k2react,k2).X_H + (k6react,k6).X_H;
AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;
AX_L = (k1react).AX_H;
B_H = (k4react,k4).B_L;
B_L = (k5react,k5)B_H + (k3react,k3).B_H;
Y_H = (k6react,k6).Y_L;
Y_L = (k3react,k3).Y_H;
```

5. Gather stoichiometric information

Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react, 2* A + X -> AX, k1;
k2react, AX -> 2* A + X, k2;
k3react, 2* AX -> B + Y, k3;
k4react, B -> 2* A, k4;
k5react, A -> B, k5;
k6react, Y -> 2* X, k6;

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

6. Include delays linked to population sizes

Rates
k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;

Species
A = 1000; X = 500;
AX = 200;
B = 0; Y = 0;

Reactions
k1react, 2*A + X -> AX, k1*min(A,X) ;
k2react, AX -> 2*A + X, k2*AX ;
k3react, 2*AX -> B + Y, k3*AX ;
k4react, B -> 2*A, k4*B ;
k5react, A -> B, k5*A ;
k6react, Y -> 2*X, k6*Y ;

k1 = 1; k2 = 1; k3 = 1;
k4 = 2*k3; k5 = 2*x; k6 = 2*b;
a = 1000; x = 500; aX = 200;

A_H = (k1react,k1).A_L + (k5react,k5).A_L;

A_L = (k2react,k2).A_H + (k4react,k4).A_H;

X_H = (k1react,k1).X_L;

X_L = (k2react,k2).X_H + (k6react,k6).X_H;

AX_H = (k2react,k2).AX_L + (k3react,k3).AX_L;

AX_L = (k1react).AX_H;

B_H = (k4react,k4).B_L;

B_L = (k5react,k5)B_H + (k3react,k3).B_H;

Y_H = (k6react,k6).Y_L;

Y_L = (k3react,k3).Y_H;

Implementation

- Module built for Choreographer, which itself is built on top of NetBeans.
- Integrate the PEPA workbench and ISBJava library.
- Modify ISBJava to allow use of min and max functions.

Modelling BitTorrent

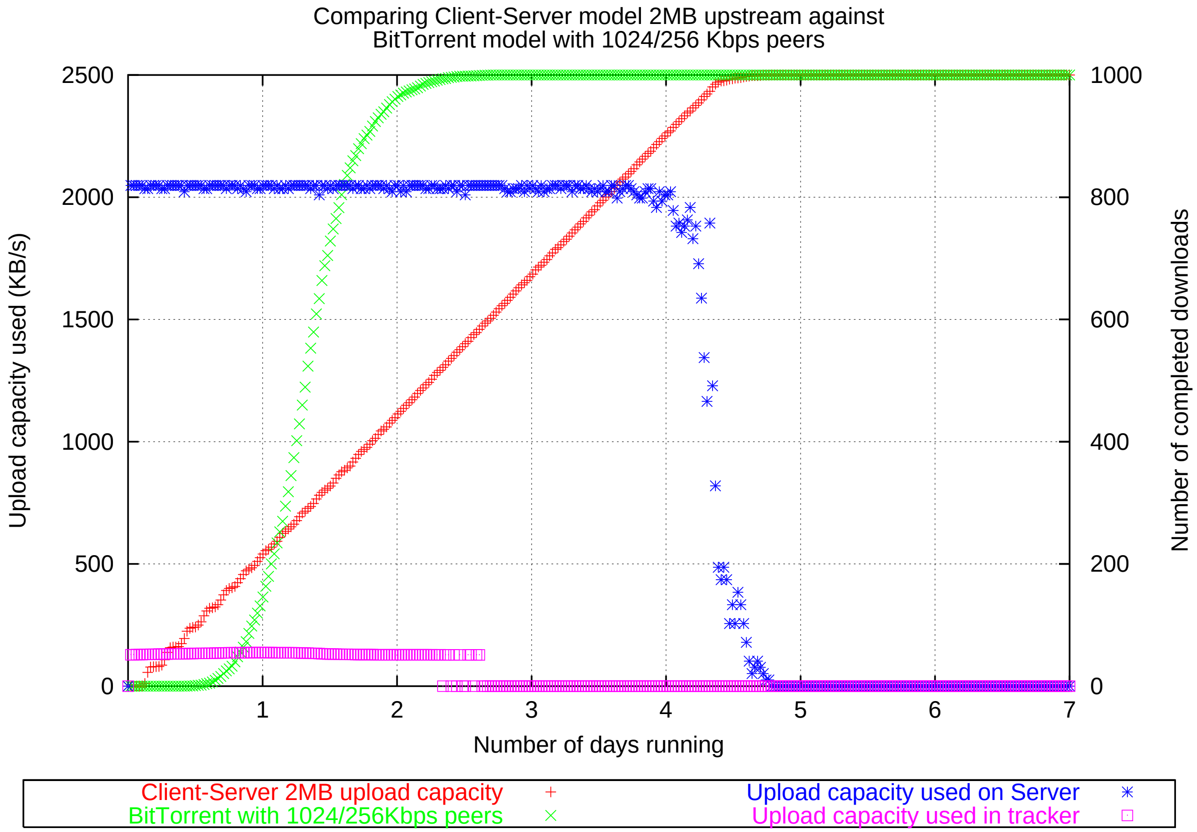
- Simple model of BitTorrent based on the excellent work of Dongyu and Srikant (fluid model).

- Expanded to more accurately measure the peers and their current status. Corrects the issue of seeds existing before it is possible.

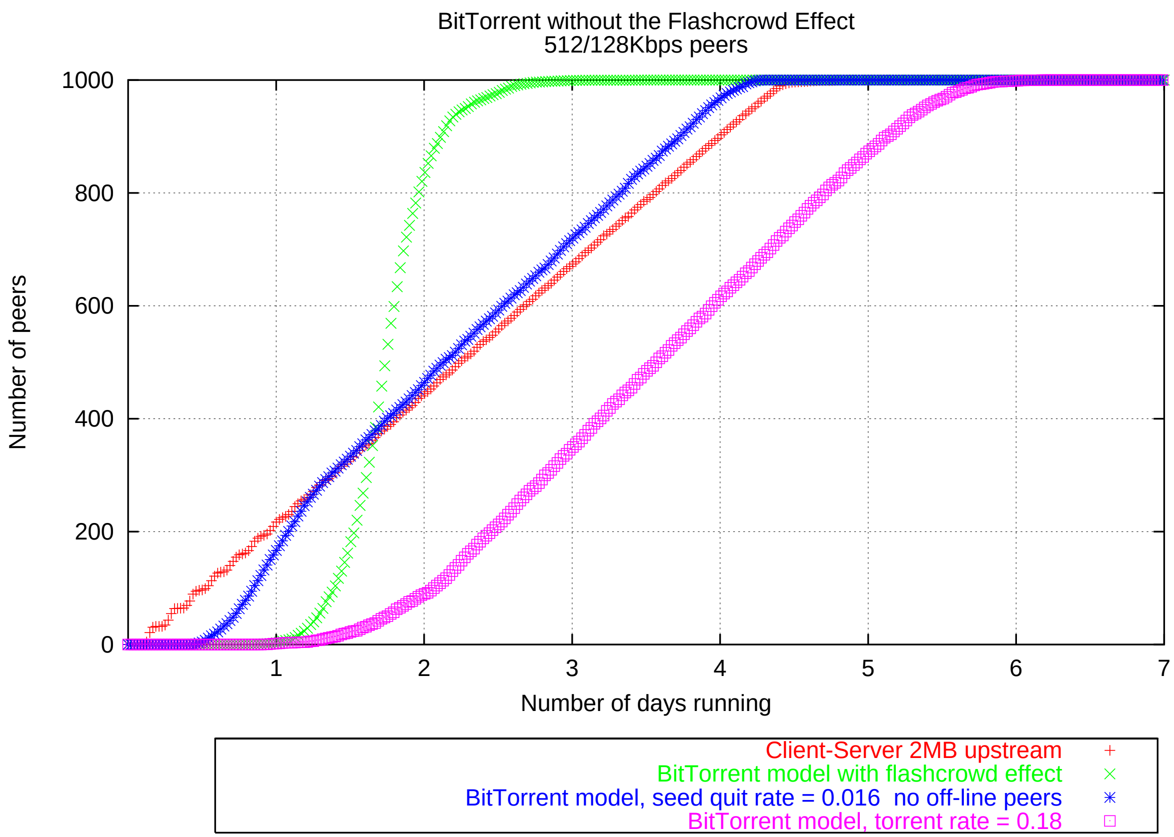
  on-line Peer is ready to download and contributing to the upload of others
  downloading Peer is downloading and contributing to the upload of others.
  off-line Peer is off-line, neither downloading or contributing.
  seed A Peer with the complete file and is currently only uploading

School of informatics

Client-Server Vs BitTorrent



Flashcrowd effect



And now on to a demonstration...