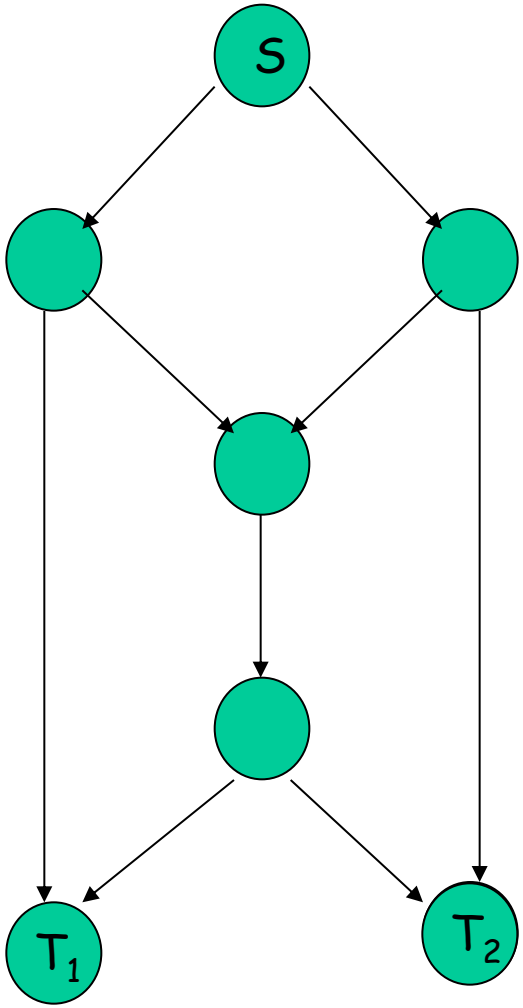
Throughput optimization …

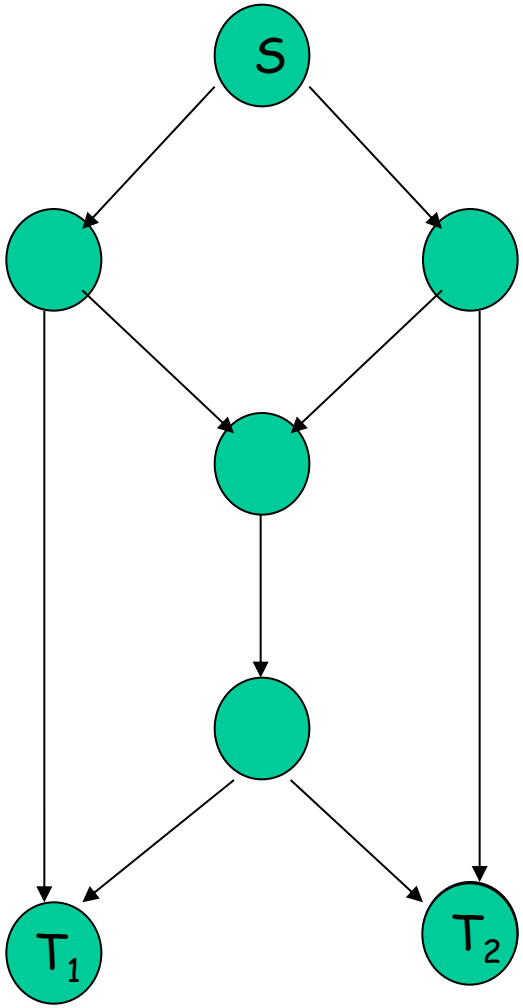
- Max-flow optimization
  - Zongpeng's slides: 2-8
- Network coding
  - Butterfly example
  - Zongpeng's slides: 14-15

Without network coding …



- Using unicast, how much can S send to T1 and T2?

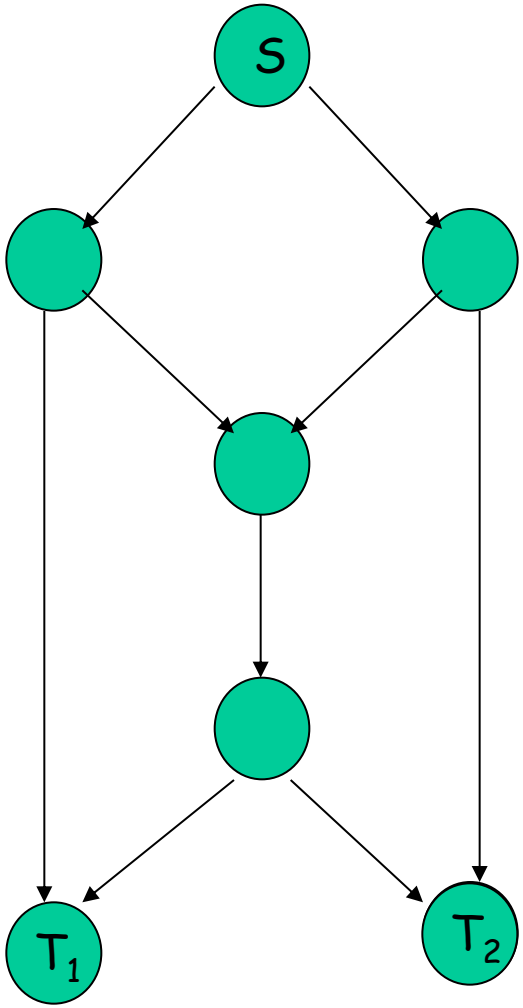
Without network coding …



- T1 and T2 both get 50% of senders capacity

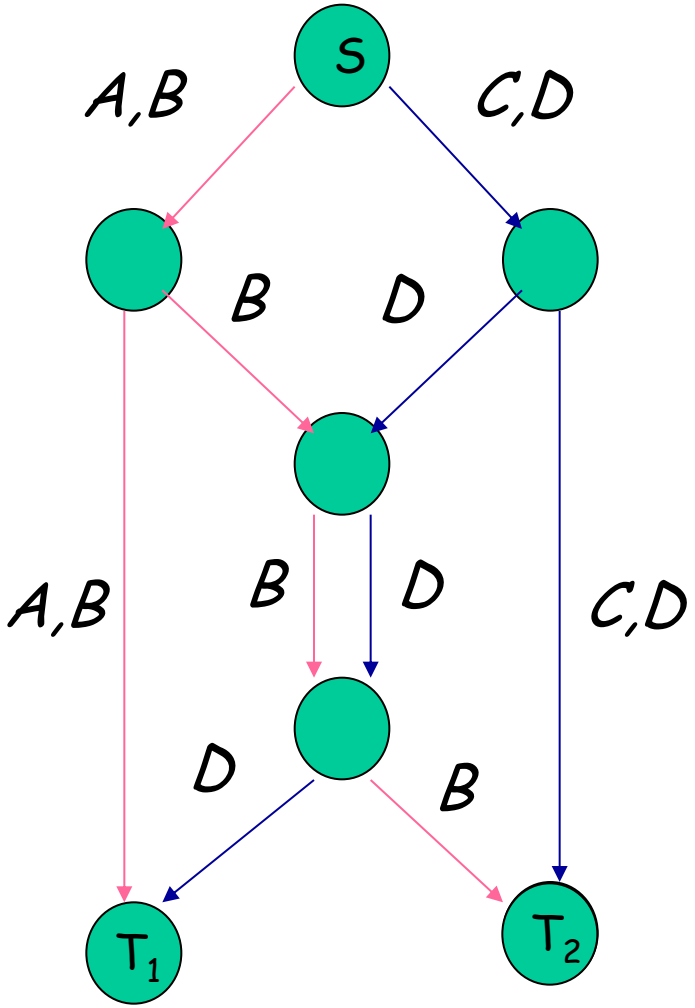
- Using unicast, how much can S send to T1 and T2?

Without network coding …



- Using multicast, how much can S send to T1 and T2?

Without network coding …

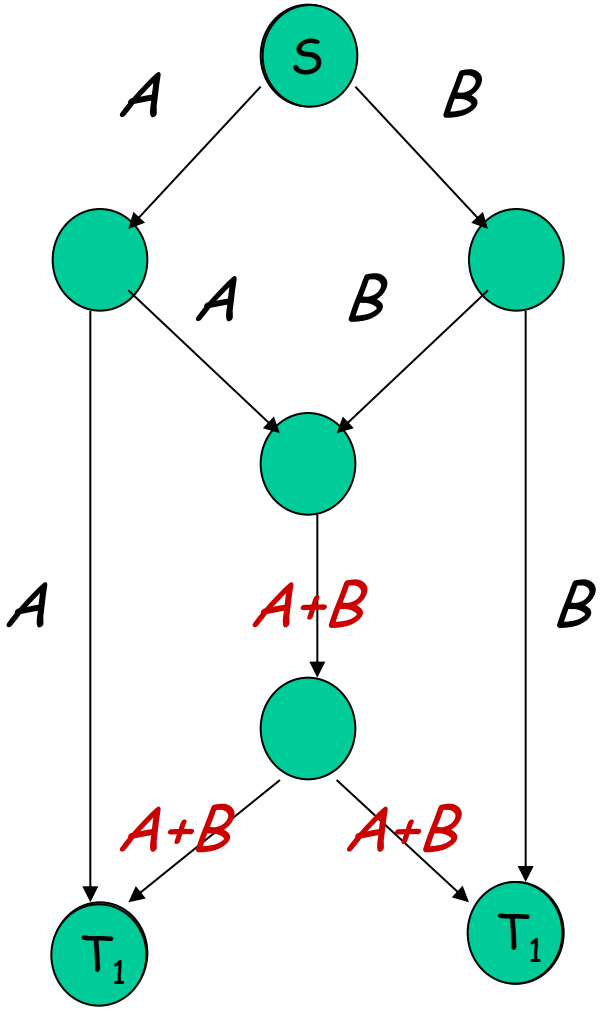


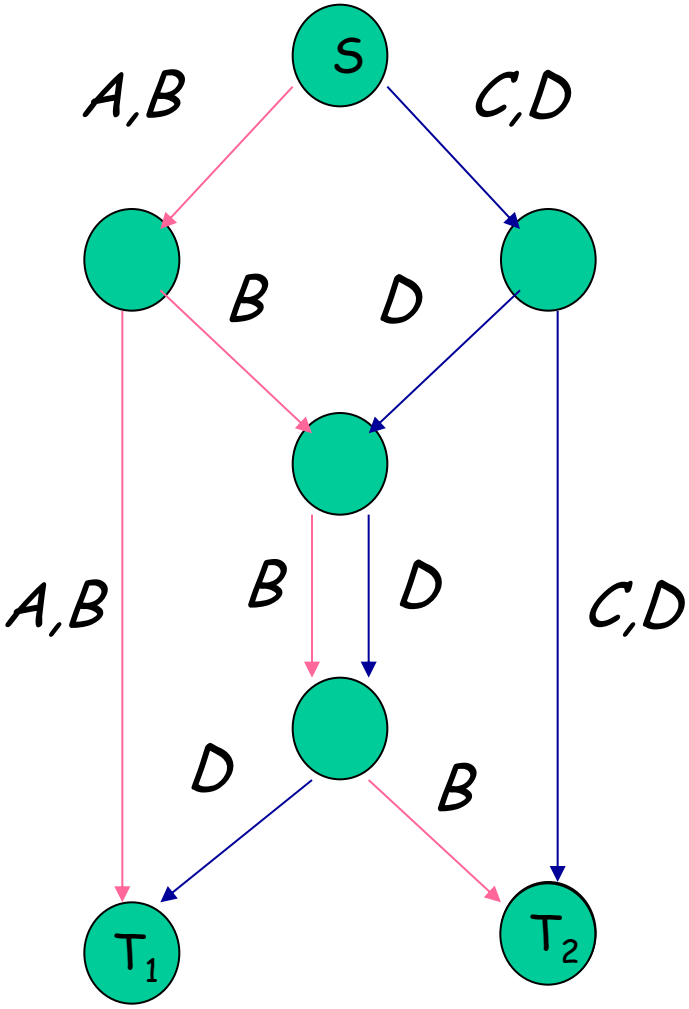
- T1 and T2 both get $\frac{3}{4}$ streams (75% of senders capacity)

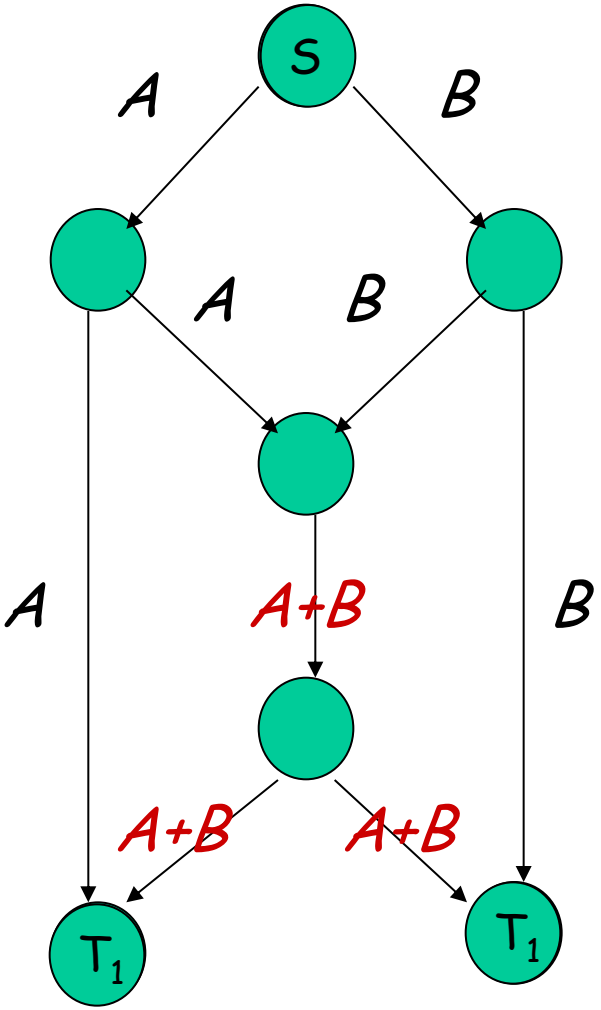
- Optimization problem equal to "packing of Steiner trees" (NP-hard problem)

With network coding …

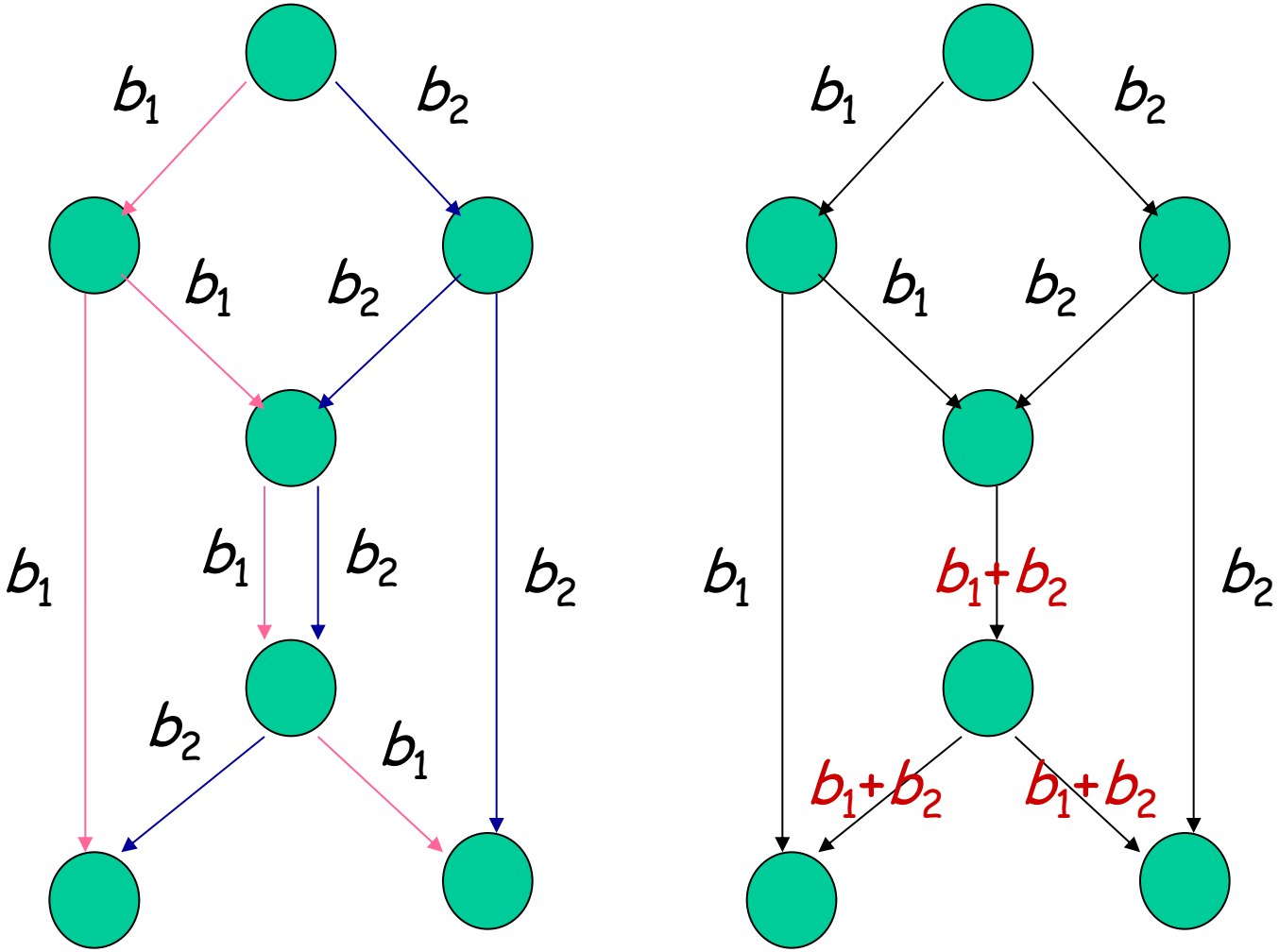
- T1 and T2 both get 2/2 streams (100% of senders capacity)
- Improvement by 33%




S
A,B
C,D
B
D
A,B
B
D
C,D
D
B
T1
T2


S
A
B
A
B
A
A+B
B
A+B
A+B
T1
T1

- Savings can also be in terms of "bandwidth"
  …


b_1
b_2
b_1
b_2
b_1
b_1
b_2
b_2
b_2
b_1
b_1
b_2
b_1+b_2
b_2
b_1+b_2
b_1+b_2

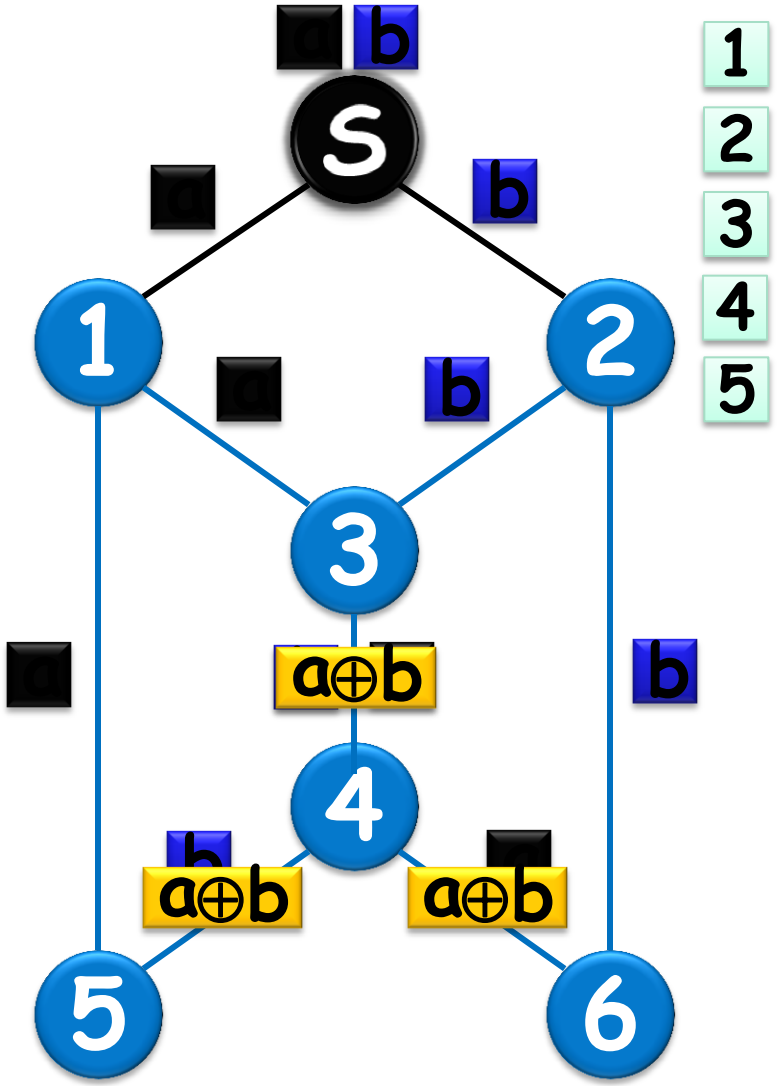
- …or “time” …

Network Coding Example (animation)

- A technique to improve:

  1. network throughput
  2. efficiency
  3. scalability
  ...

- Information is coded at potentially every node

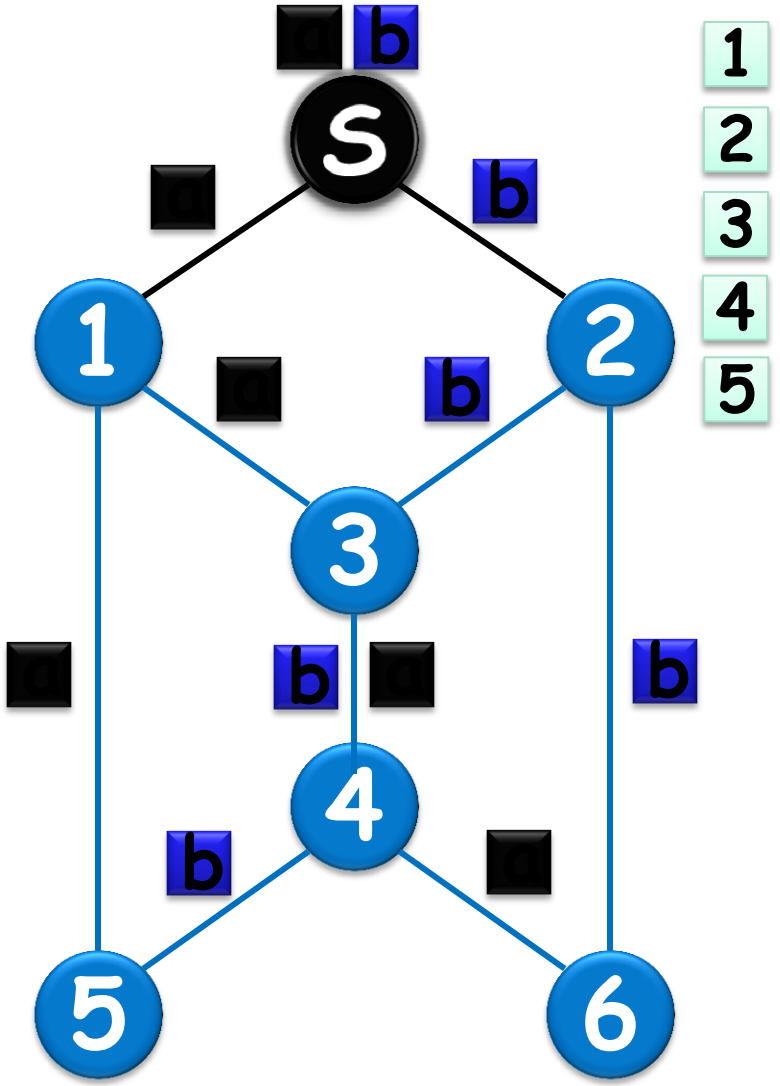


Network Coding Example (without)

- A technique to improve:

  1. network throughput
  2. efficiency
  3. scalability
  ...

- Information is **coded** at potentially every node

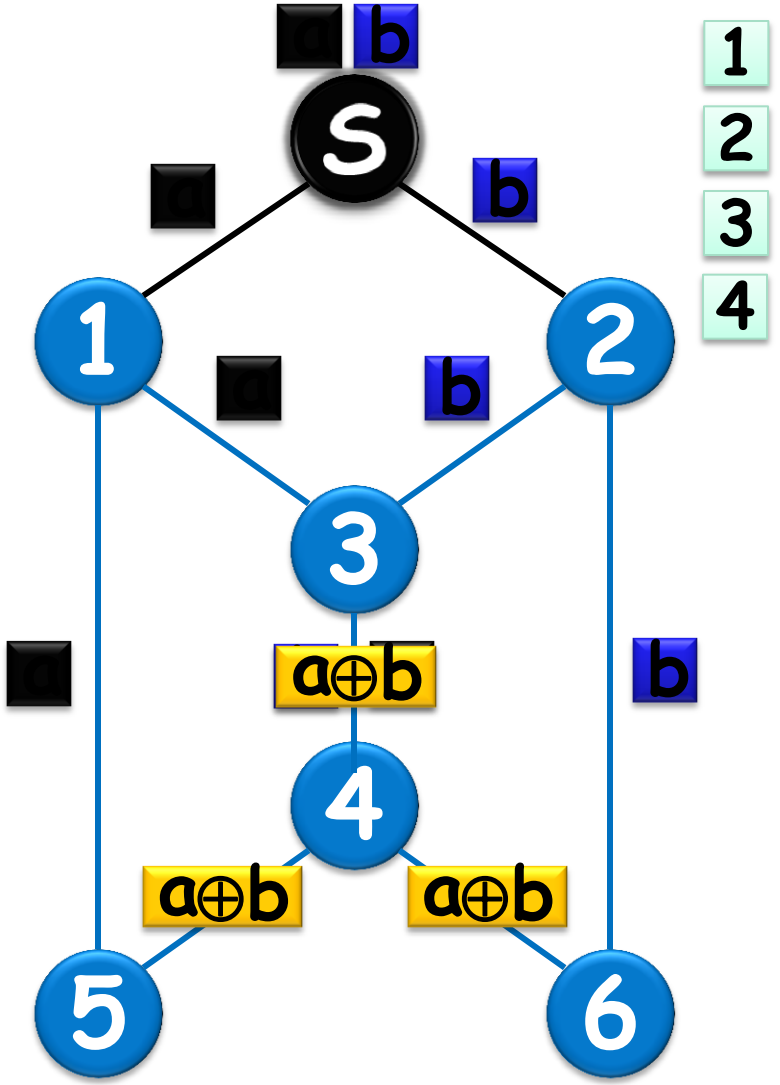


Network Coding Example (with)

- A technique to improve:

  1. network throughput
  2. efficiency
  3. scalability
  ...

- Information is coded at potentially every node

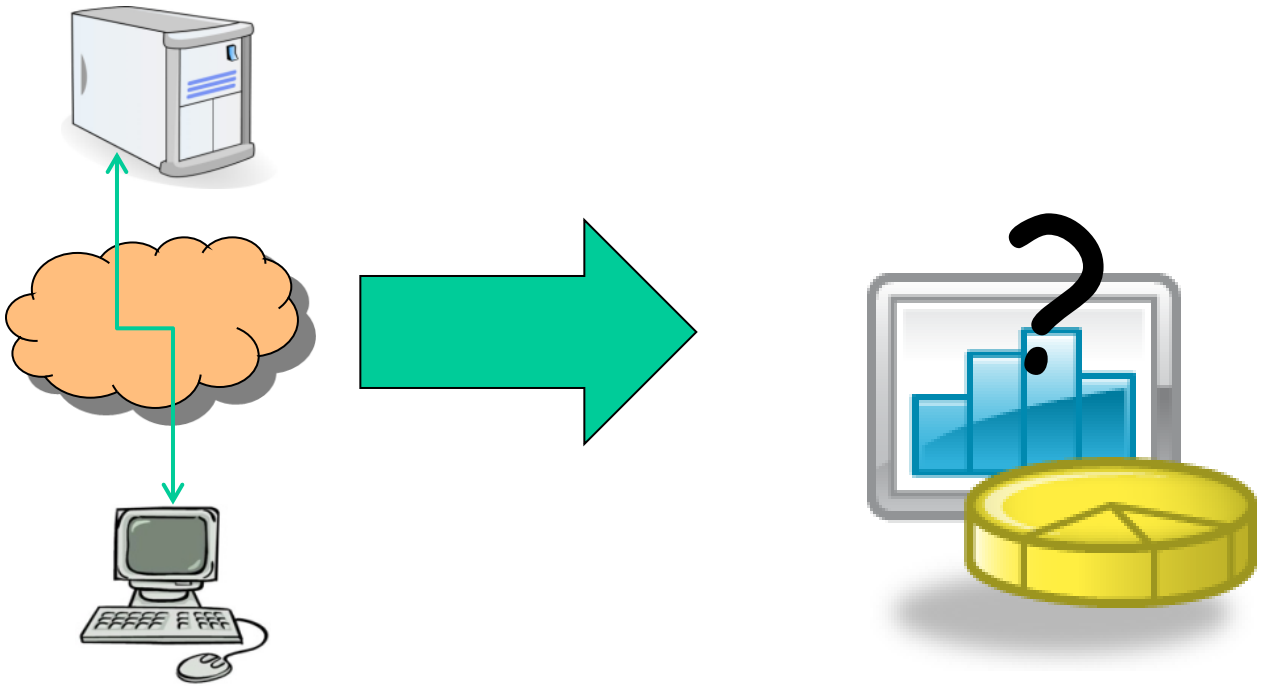


6.1: Cache-size discussion

- Consider first the workloads that these caches will need to serve …

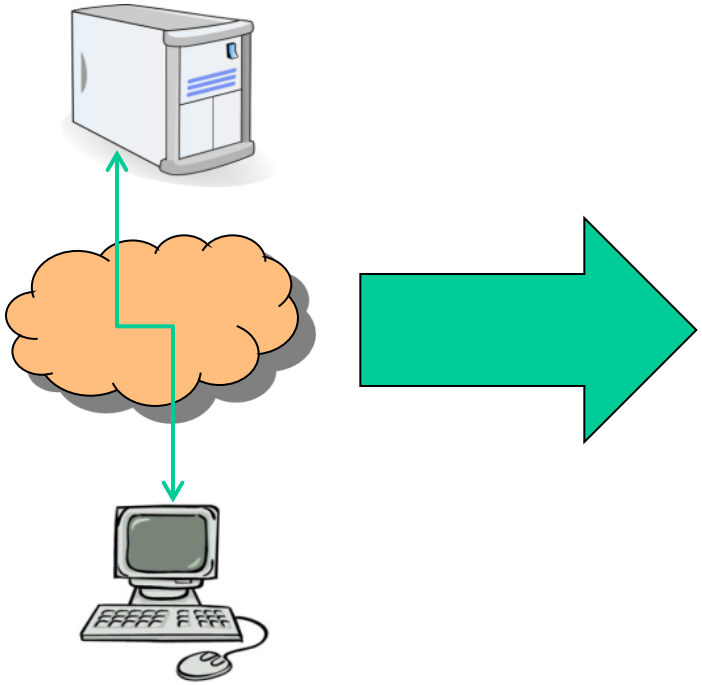
6.1: Cache-size discussion

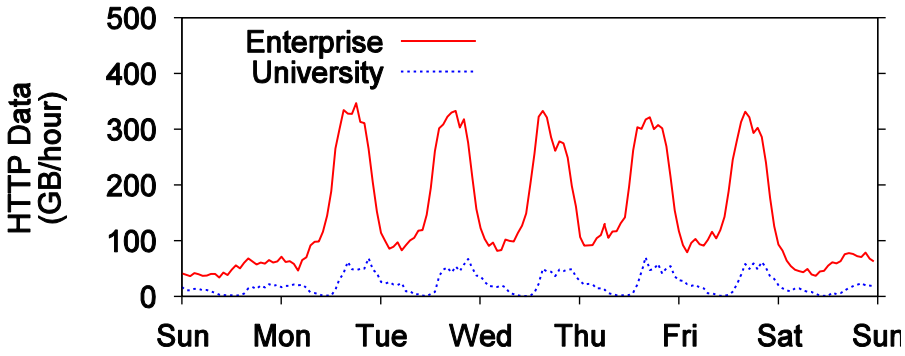
- Consider first the workloads that these caches will need to serve …



Request patterns

- Some research slides …
  - Diurnal patterns
  - Geographic differences

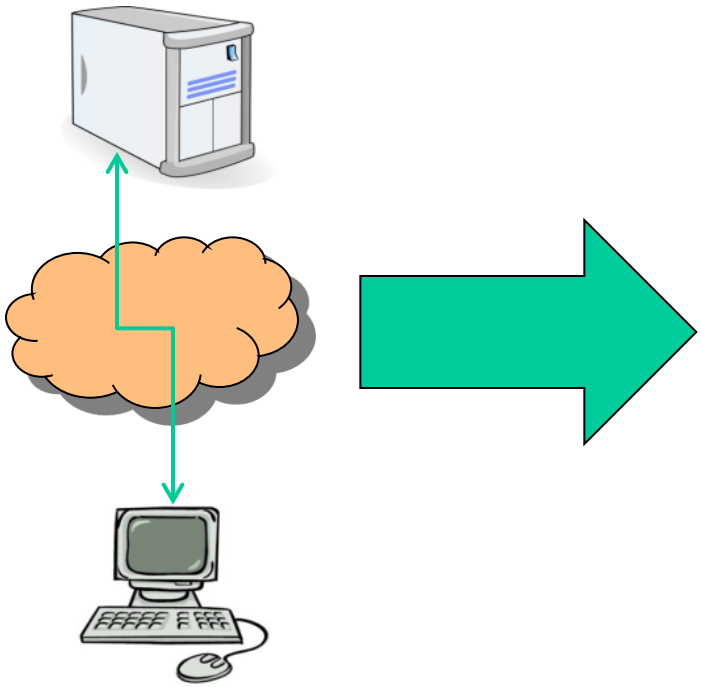


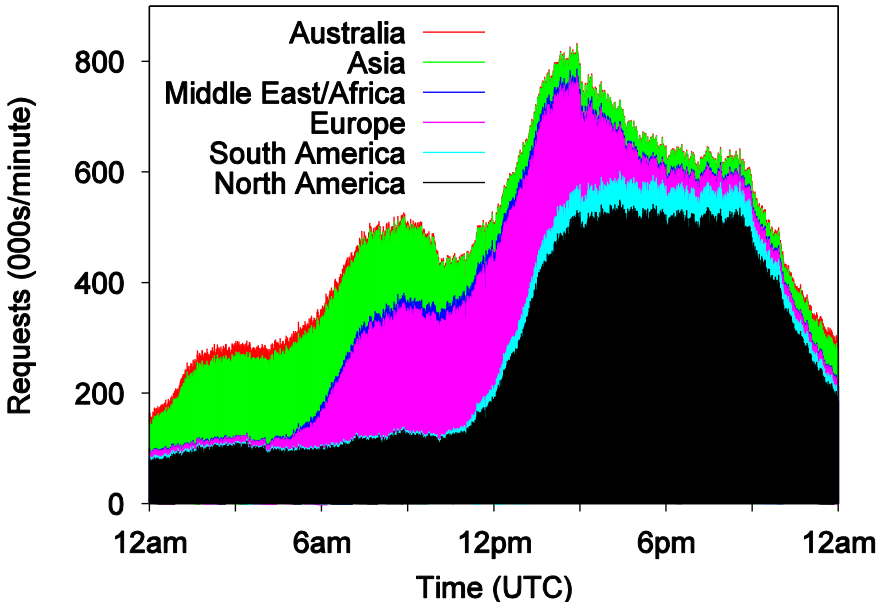


E.g., ACM TWEB

Request patterns

- Some research slides …
  - Diurnal patterns
  - Geographic differences

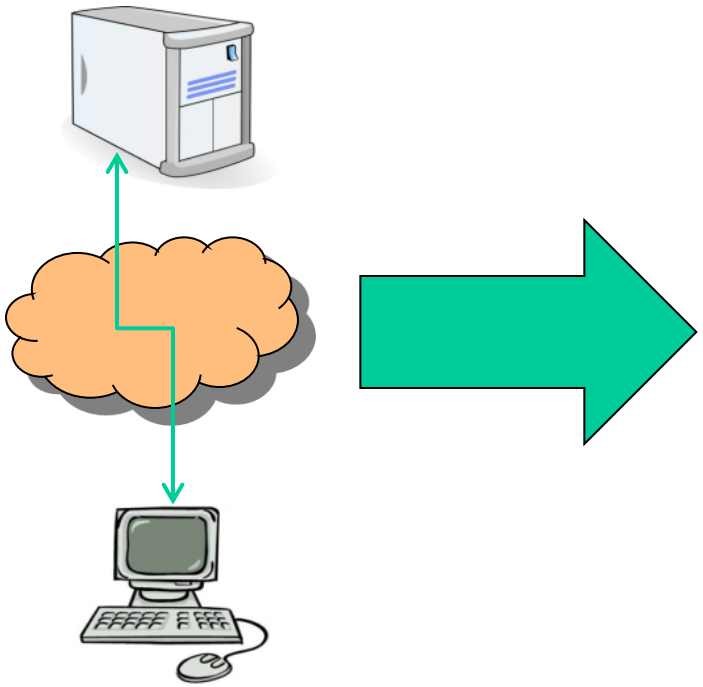


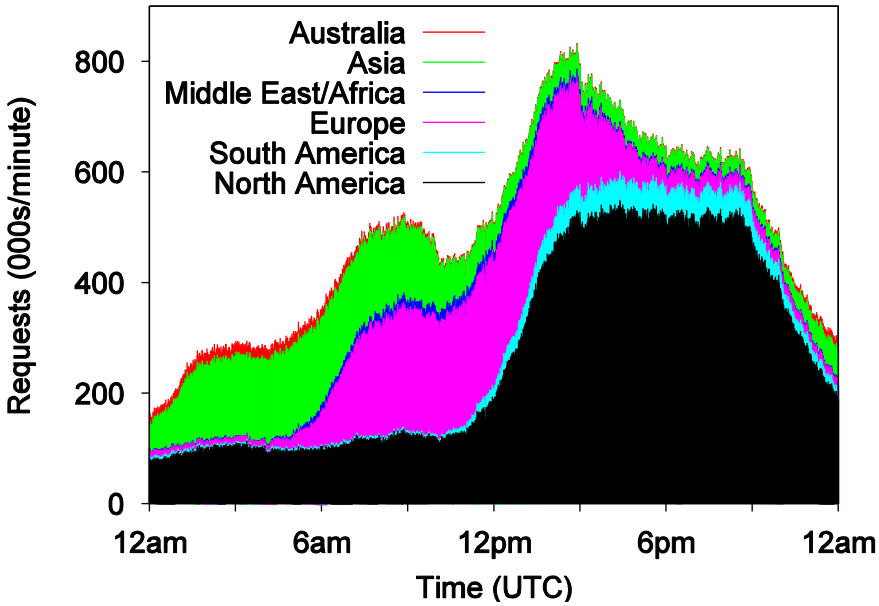


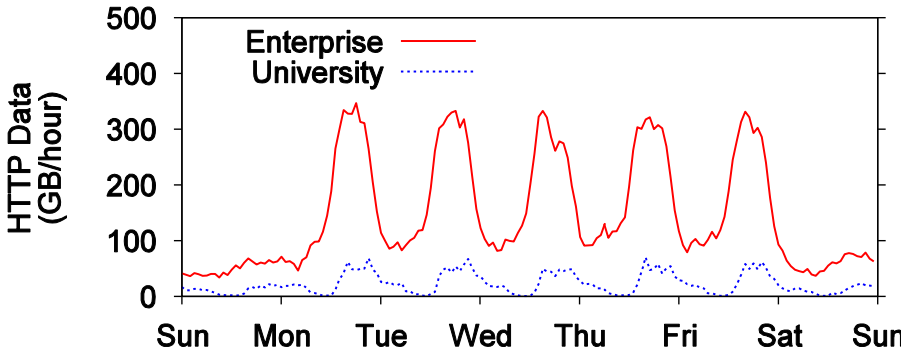
E.g., ACM TWEB, IEEE ICC '12

Request patterns

- Some research slides …
  - Diurnal patterns
  - Geographic differences



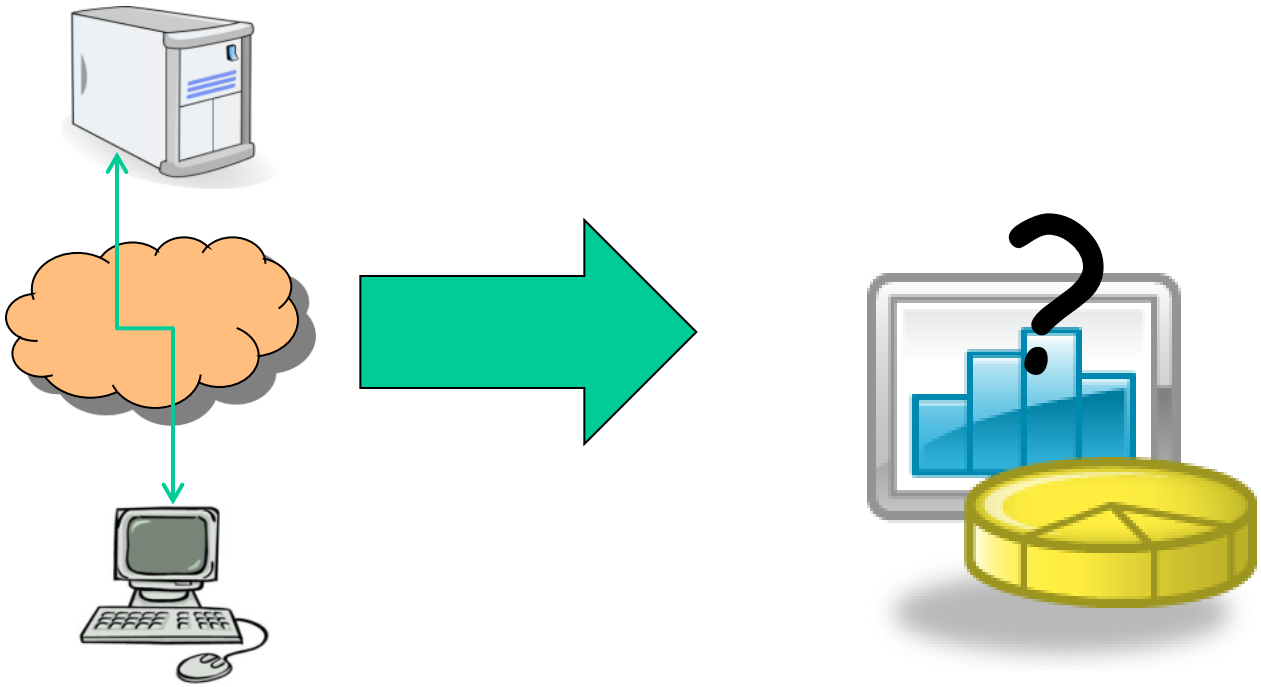




E.g., ACM TWEB, IEEE ICC '12

Request pattern …

- E.g., request inter-arrival times
  - Example 1 …



At a smaller time scale …

- Often break into stationary periods …
  - Common (but in some cases highly debated) assumption: **Poisson request process** (independent request arrivals, with exponential inter-arrival time distribution)
  - Remember: nice properties such as "memory less" and PASTA (Poisson Arrivals See Time Average)

File popularity distribution and "heavy" tails

- Example slides with YouTube popularity
  - but web object popularity, file size distributions, number of friends in social networks, etc. often see similar "heavy tail" distributions ...
  - Should note (this list can be made very very long, and include things such as the frequency words are used, the size of cities, the size of earthquakes, the size of bacteria cultures … and the list will go on ... and on … and on ...

Motivation

- Video dissemination (e.g., YouTube) can have wide-spread impacts on opinions, thoughts, and cultures

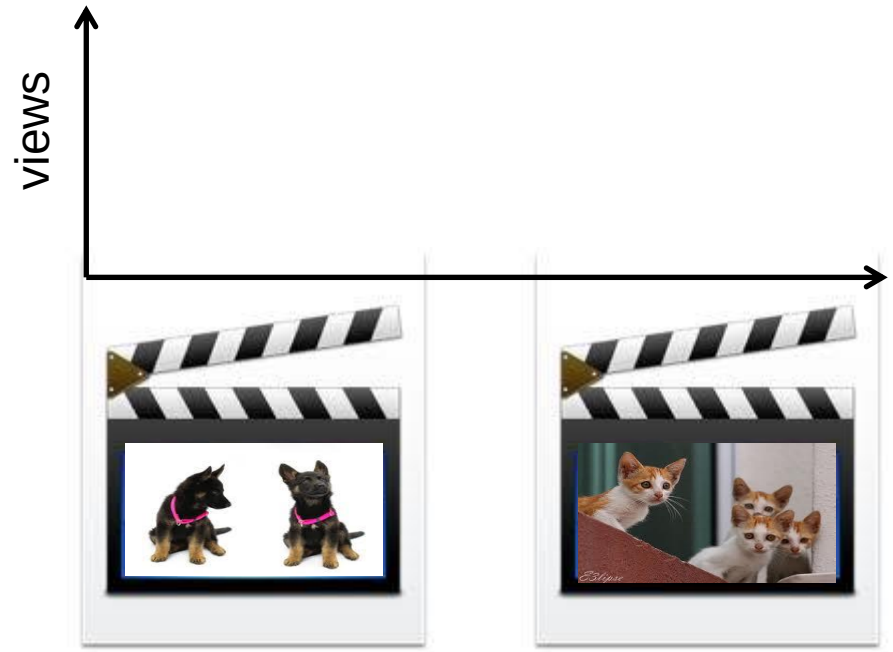
E.g., ACM KDD '12, IFIP Performance '11, ACM TWEB

Motivation

- Not all videos will reach the same popularity and have the same impact

E.g., ACM KDD '12, IFIP Performance '11, ACM TWEB

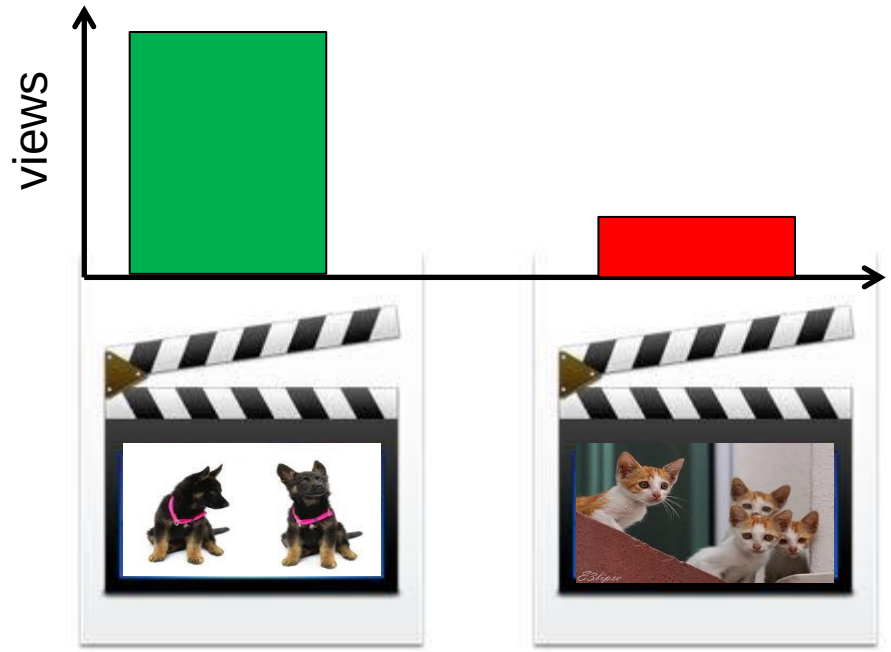
Motivation



- Not all videos will reach the same popularity and have the same impact

E.g., ACM KDD '12, IFIP Performance '11, ACM TWEB

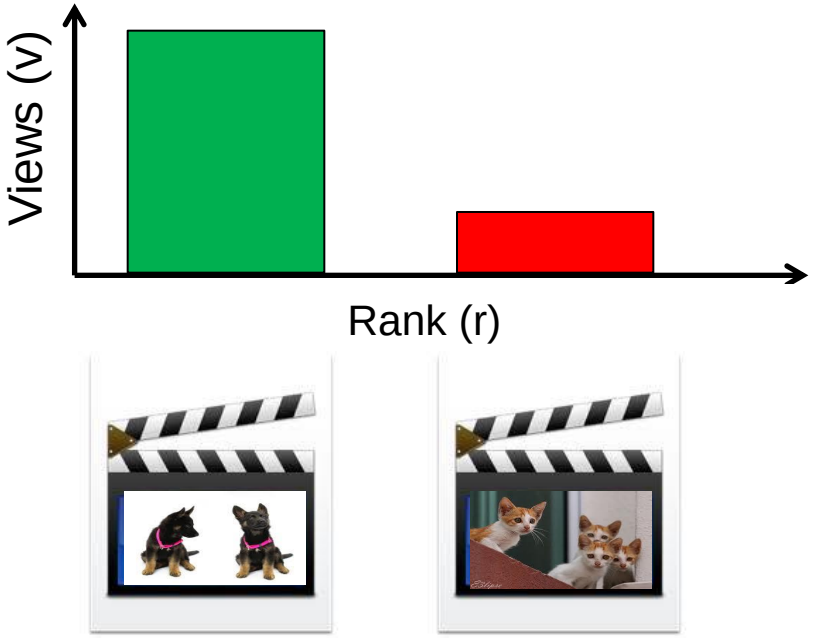
Motivation



- Not all videos will reach the same popularity and have the same impact

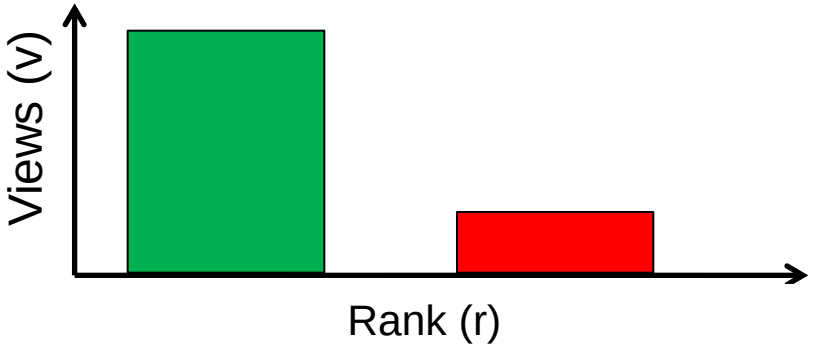
E.g., ACM KDD '12, IFIP Performance '11, ACM TWEB

Popularity distribution



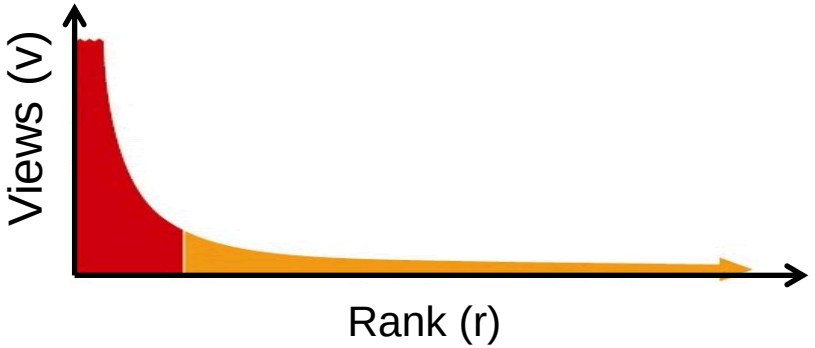
E.g., ACM KDD ‘12

Popularity distribution

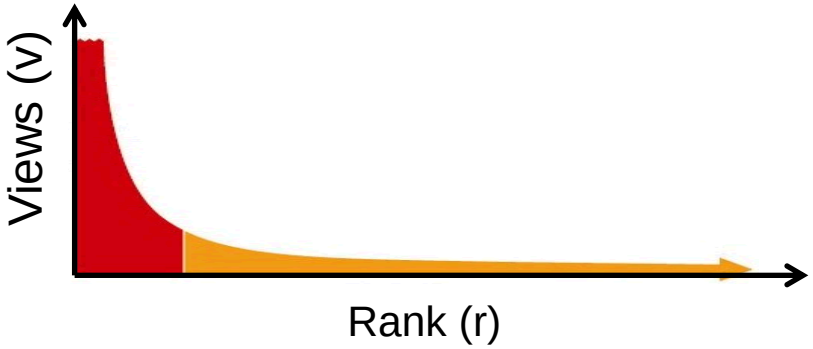


E.g., ACM KDD ‘12

Popularity distribution

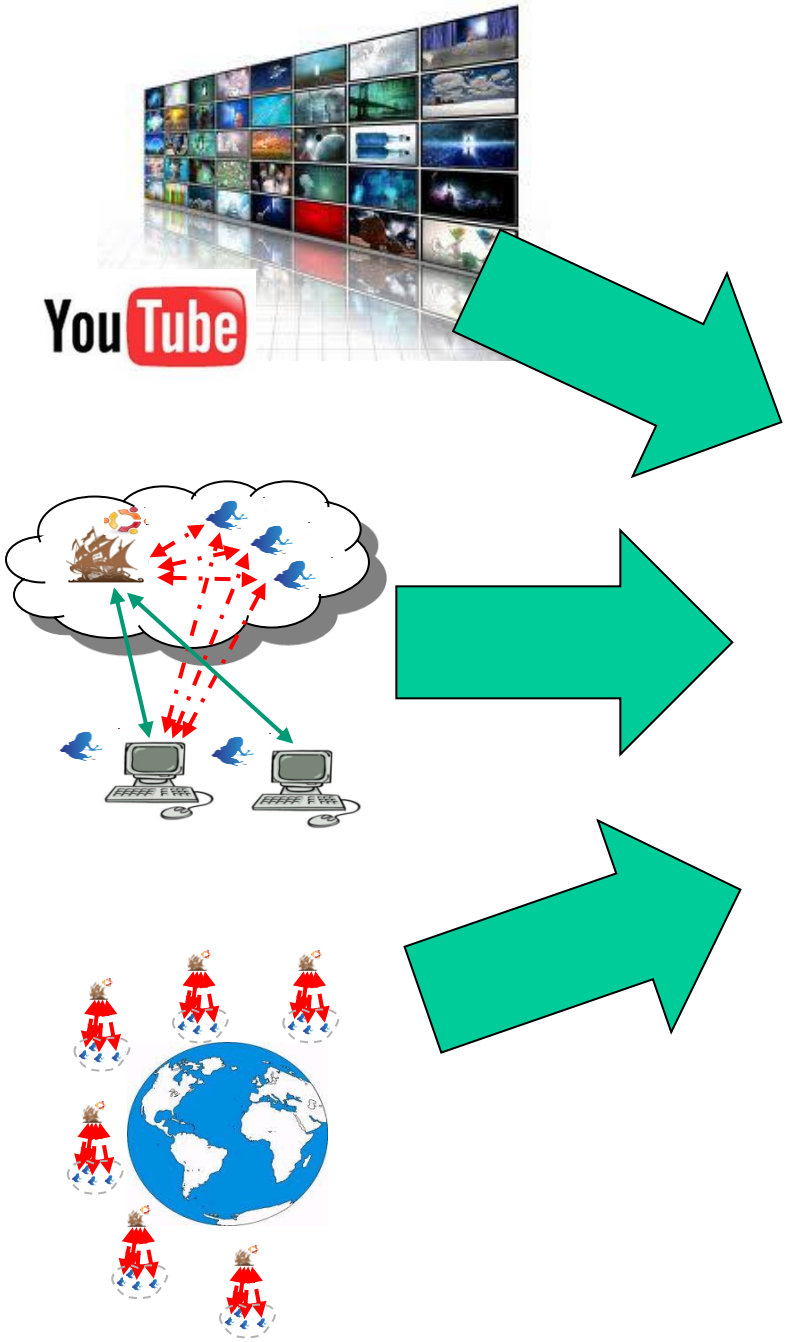


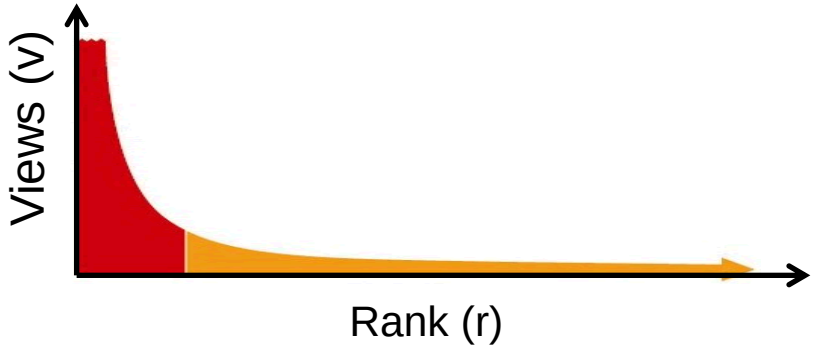
Popularity distribution



E.g., ACM KDD ’12, PAM ’12

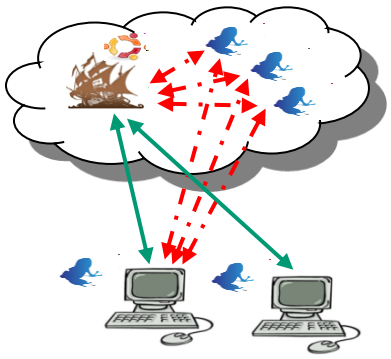
Popularity distribution

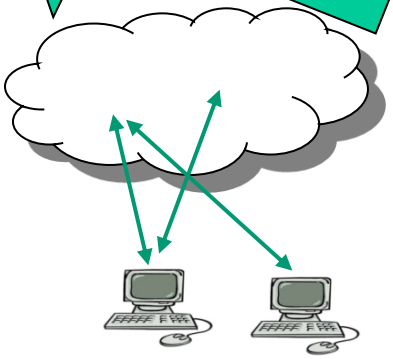


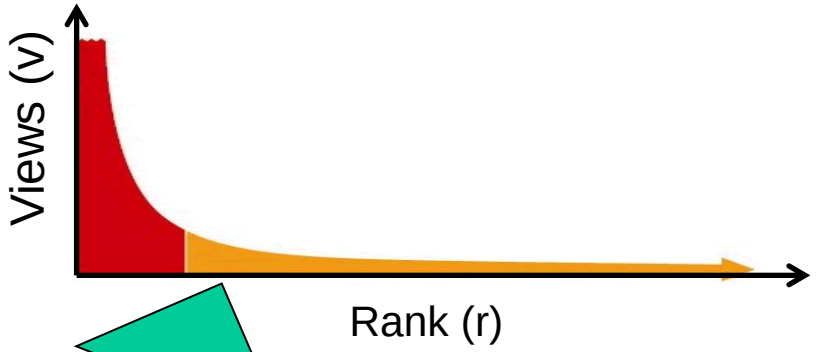


E.g., ACM KDD ’12, PAM ’12

Popularity distribution





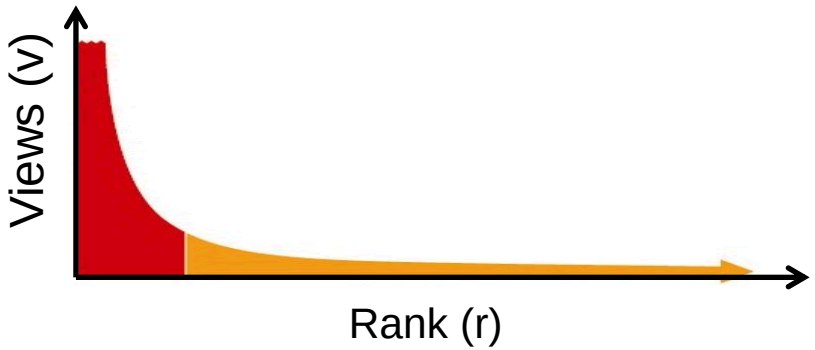


E.g., ACM KDD ’12, PAM ’12, ACM TWEB

Let's look at an example …

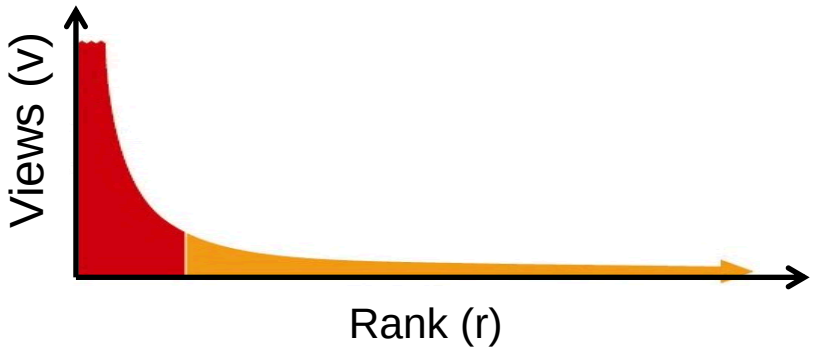
- Example 2

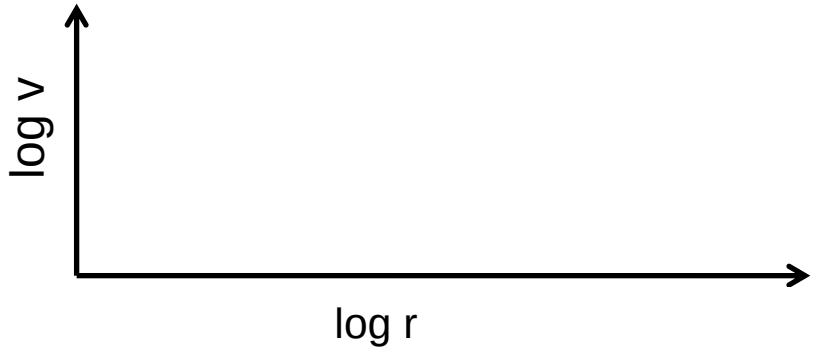
Zipf popularity...
... and long tails



$$v_r \propto r^{-\alpha}$$

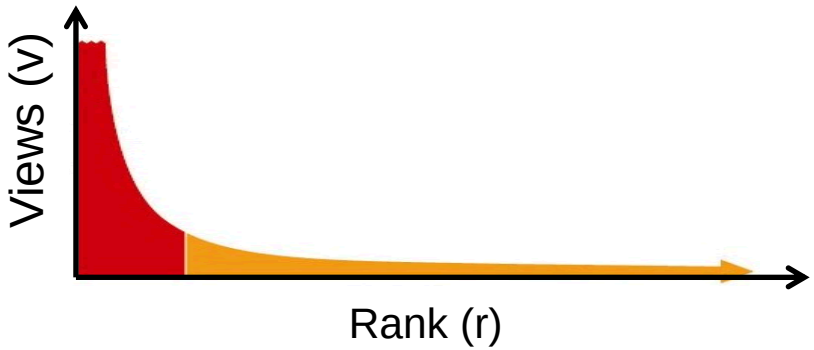
Zipf popularity...
... and long tails

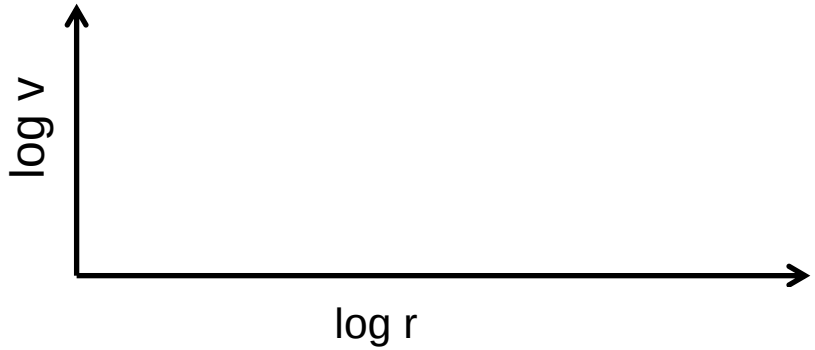




$$v_r \propto r^{-\alpha}$$

Zipf popularity...
... and long tails

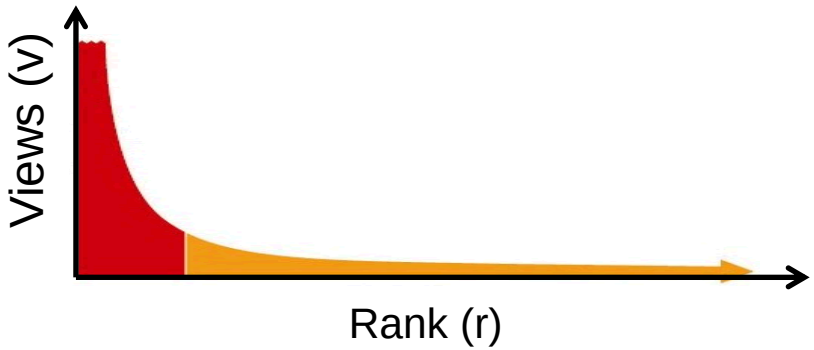


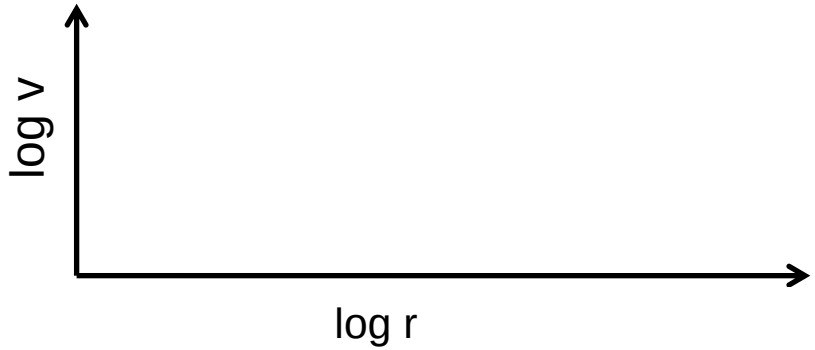


$$v_r \propto r^{-\alpha}$$

$$\log v_r = \log v_1 - \alpha \log r$$

Zipf popularity...
... and long tails

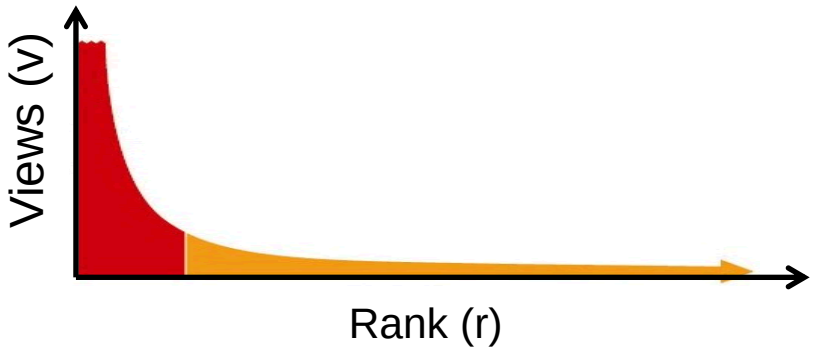


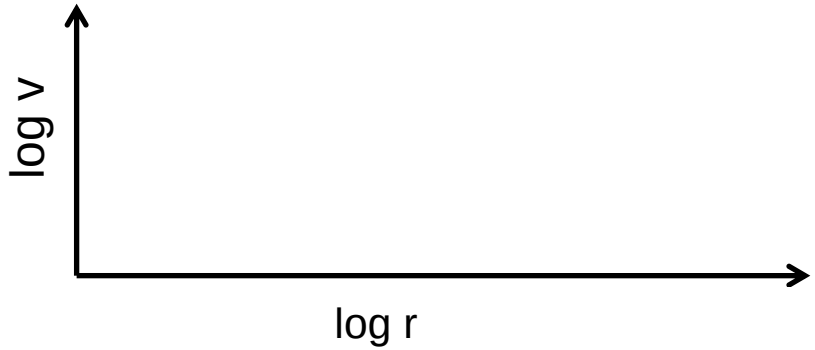


$$v_r \propto r^{-\alpha}$$

$$\log v_r = \log v_1 - \alpha \log r$$

Zipf popularity...
... and long tails



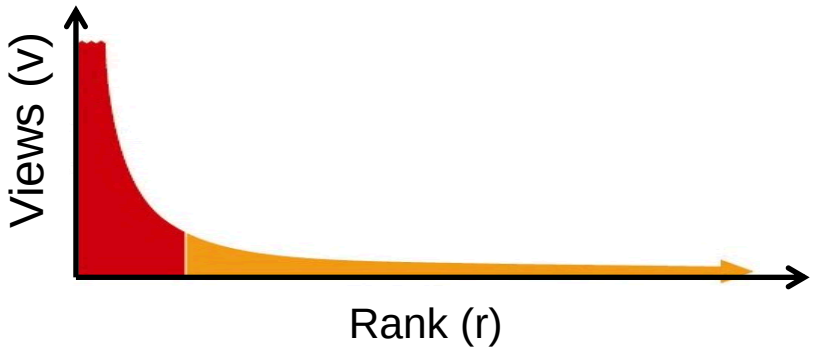


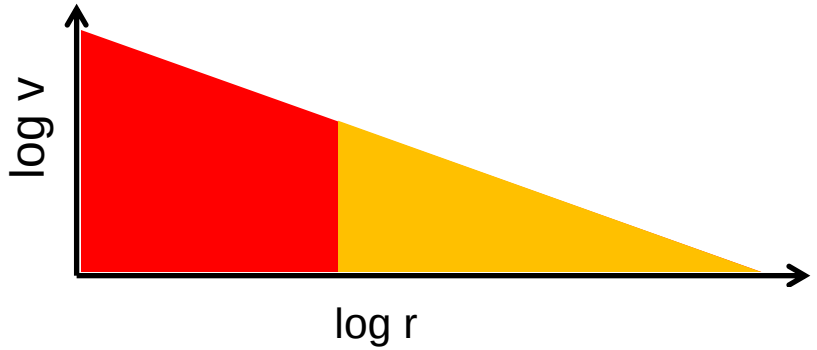
$$v_r \propto r^{-\alpha}$$

$$\log v_r = \log v_1 - \alpha \log r$$

y(x) = x_0 -- α x

Zipf popularity...
... and long tails



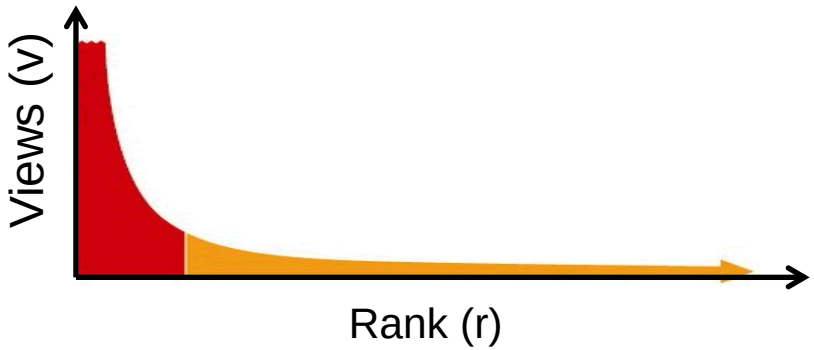


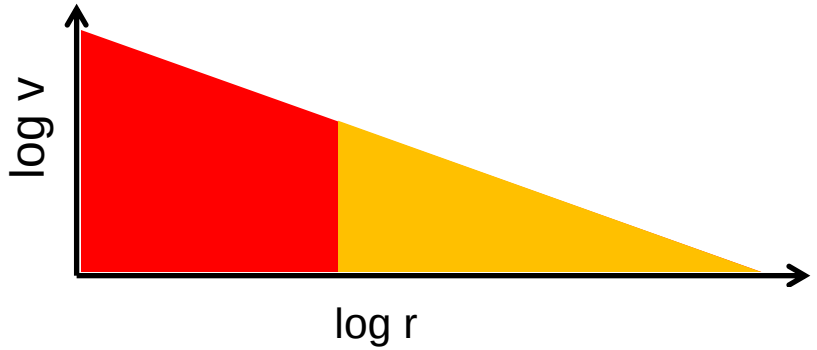
$$v_r \propto r^{-\alpha}$$

$$\log v_r = \log v_1 - \alpha \log r$$

y(x) = x_0 -- α x

Zipf popularity...
... and long tails

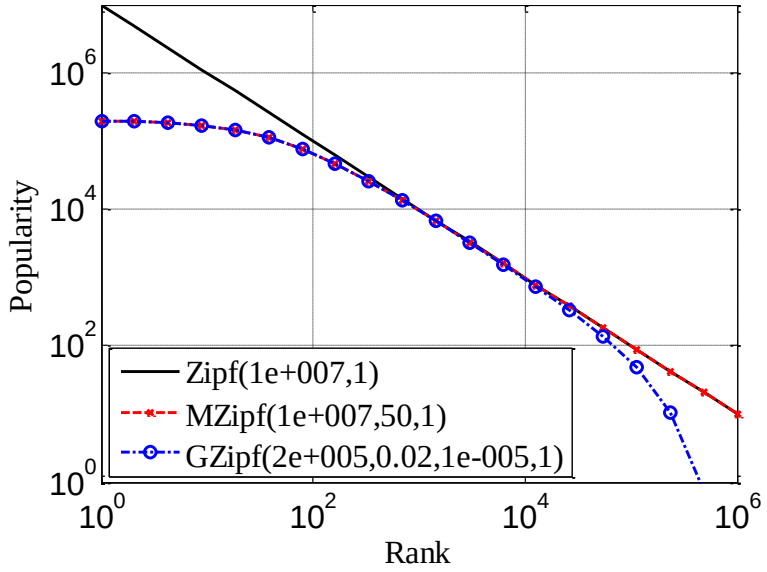


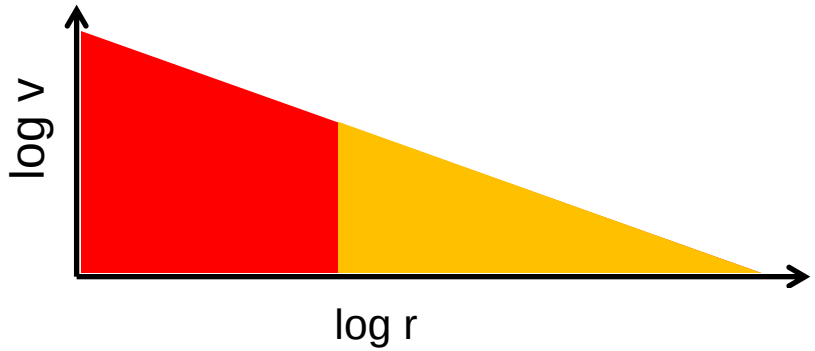


$$v_r \propto r^{-\alpha}$$

$$\log v_r = \log v_1 - \alpha \log r$$

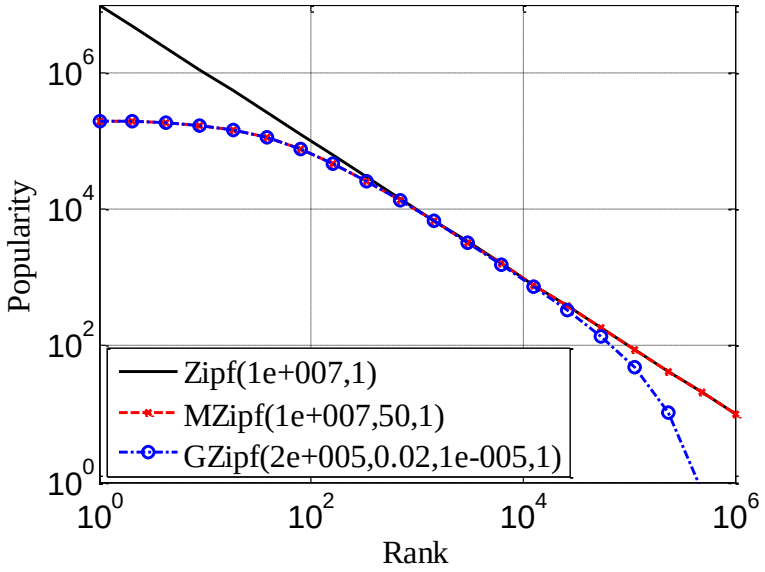
Zipf popularity…
… and long tails





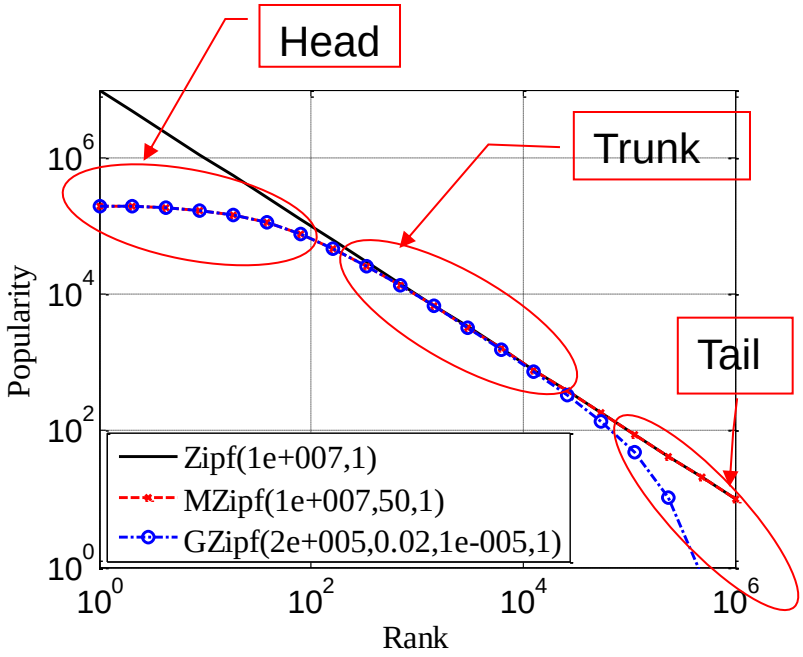
$$\log v_r = \log v_1 - \alpha \log r$$

Zipf popularity…
… and long tails



E.g., ACM TWEB, PAM ‘11
IFIP Performance ‘11, IPTPS ‘10

Zipf popularity… … and long tails



- Popularity distribution statistics
  - Across services (impact on system design)
  - Lifetime vs current
  - Over different time period (churn)
  - Different sampling methods

E.g., ACM TWEB, PAM ’11, IFIP Performance ‘11, IPTPS ‘10

Heavy-tail distributions …

- "A probability distribution is said to have a heavy tail if the tail is not exponentially bounded"
  - E.g., paper and references therein: "A Tale of the Tails: Power-laws in Internet Measurements", IEEE Network, Mahanti et al., 2013
- Power-law, Pareto, Zipf (in some sense the same)
- … and then there are many many "heavy tail" distributions, variations and generalizations, including distributions such as log-normal, various generalized Zipf/Pareto distributions, etc.

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i =

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) =

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) = $\exp(-\lambda_i T)$

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) = $\exp(-\lambda_i T)$
  - Total miss rate =

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) = $\exp(-\lambda_i T)$
  - Total miss rate = $\Sigma_i \lambda_i \exp(-\lambda_i T)$

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) = $\exp(-\lambda_i T)$
  - Total miss rate = $\Sigma_i \lambda_i \exp(-\lambda_i T)$
  - (normalized) hit rate =

Let's build a (very) simple model

- Poisson request process with request rate λ
- N objects; equal size L; Zipf popularity distribution with shape parameter α
  - Each request has request frequency λ_i = ??
  - What is the (normalized) hit rate if cache can store n objects?
- Approximate large LRU-cache with a TTL-cache
  - T = average time (without request) before cache eviction
- PASTA (Poisson Arrivals See Time Average)
  - Miss rate object i = λ_i Pr(no request in last T)
  - Pr(no request in last T) = $\exp(-\lambda_i T)$
  - Total miss rate = $\Sigma_i \lambda_i \exp(-\lambda_i T)$
  - (normalized) hit rate = $1 - (\Sigma_i \lambda_i \exp(-\lambda_i T))/(\Sigma_i \lambda_i)$

Let's build a (very) simple model

- Expected time object i in cache?

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) =

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) =

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) =

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)
- Solve
  - (time in system) =

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)
- Solve
  - (time in system) = (# in system) / (rate in/out)
    $= (1- \exp(-\lambda_i T)) / \lambda_i \exp(-\lambda_i T)$
    $= (1/\lambda_i)(\exp(\lambda_i T)-1)$

Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)
- Solve
  - (time in system) = (# in system) / (rate in/out)
    $= (1- \exp(-\lambda_i T)) / \lambda_i \exp(-\lambda_i T)$
    $= (1/\lambda_i)(\exp(\lambda_i T)-1)$
- Sanity check ...

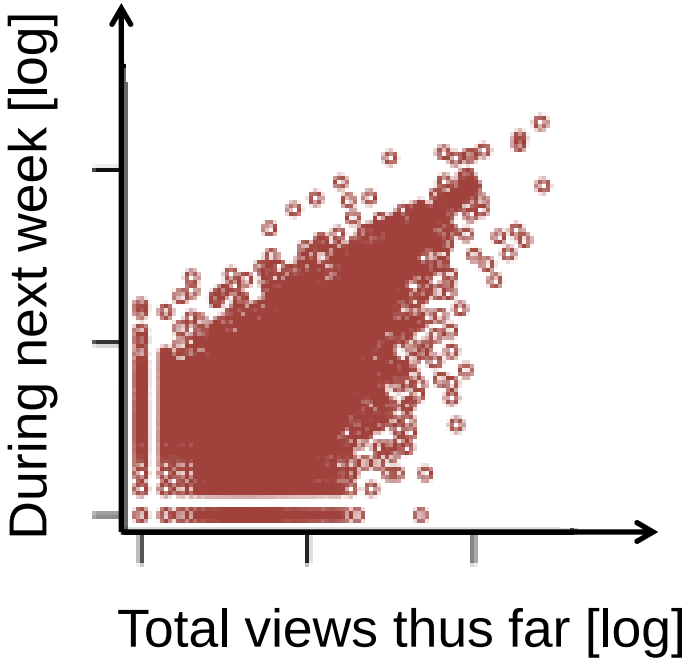
Let's build a (very) simple model

- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)
- Solve
  - (time in system) = (# in system) / (rate in/out)
    $= (1- \exp(-\lambda_i T)) / \lambda_i \exp(-\lambda_i T)$
    $= (1/\lambda_i)(\exp(\lambda_i T)-1)$
- Sanity check ...
  - Taylor expansion … (e.g., look at small and large λ_i)

Let's build a (very) simple model

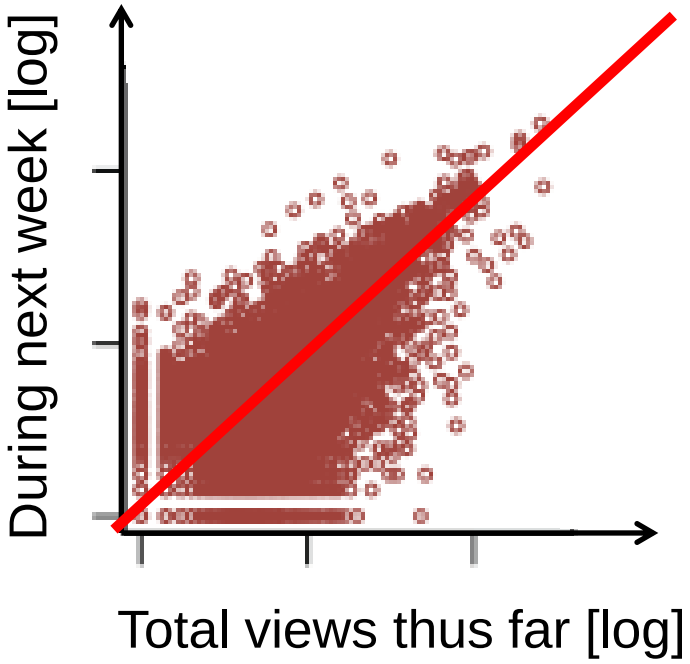
- Expected time object i in cache?
- Little's law [averages]:
  - (# in system) = (rate in/out) x (time in system)
- Identify in system
  - (rate in/out) = (miss rate object i) = $\lambda_i \exp(-\lambda_i T)$
  - (# in system) = (copies of object i in the cache) = $(1- \exp(-\lambda_i T))$
  - (time in system) = (time object i is in cache until evicted)
- Solve
  - (time in system) = (# in system) / (rate in/out)
    $= (1- \exp(-\lambda_i T)) / \lambda_i \exp(-\lambda_i T)$
    $= (1/\lambda_i)(\exp(\lambda_i T)-1)$
- Sanity check ...
  - Taylor expansion ... (e.g., look at small and large λ_i)
  - Busy period for a M/D/∞

Rich-gets-richer …
… and churn



E.g., IFIP Performance ‘11

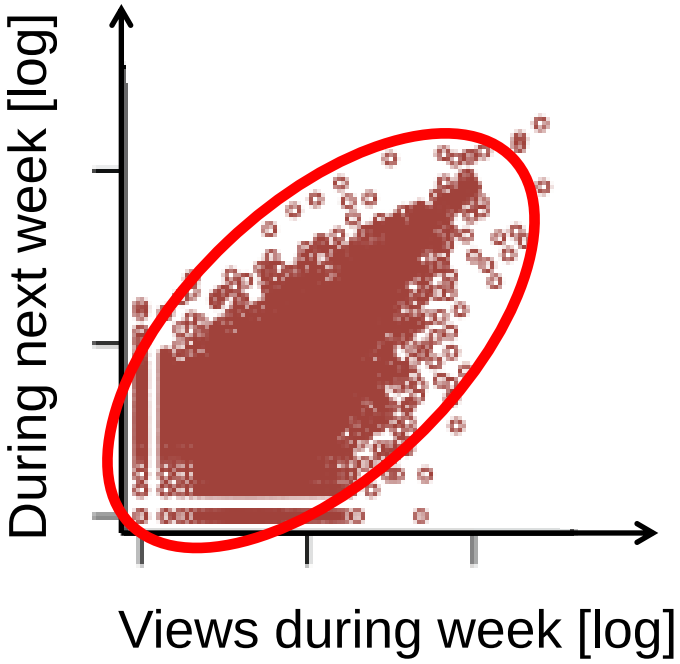
Rich-gets-richer …
… and churn



- The more views a video has, the more views it is likely to get in the future

E.g., IFIP Performance '11

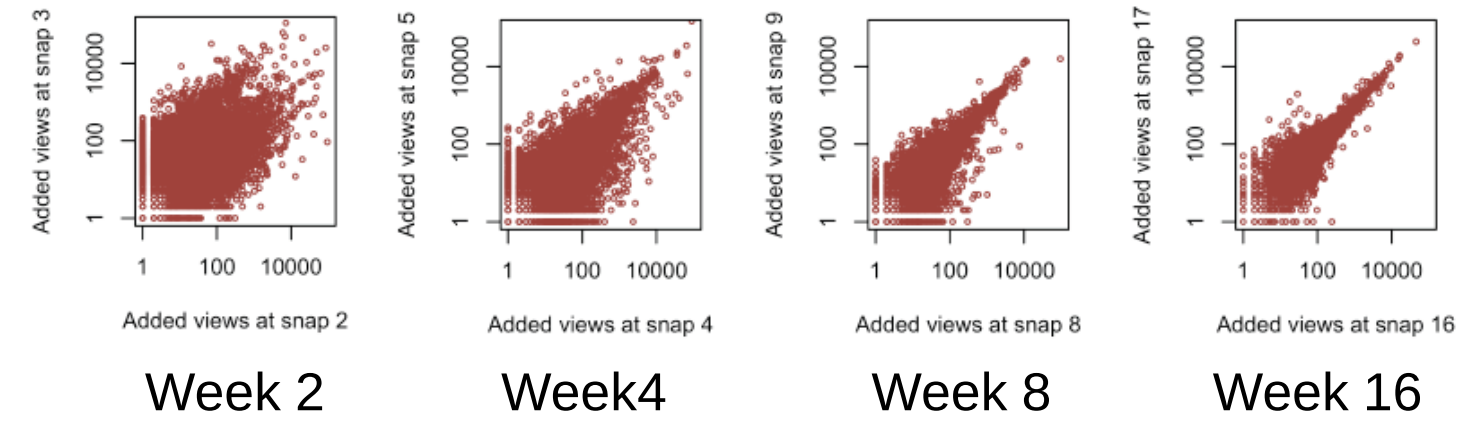
Rich-gets-richer …
… and churn



- The more views a video has, the more views it is likely to get in the future
- The relative popularity of the individual videos are highly non-stationary

E.g., IFIP Performance '11

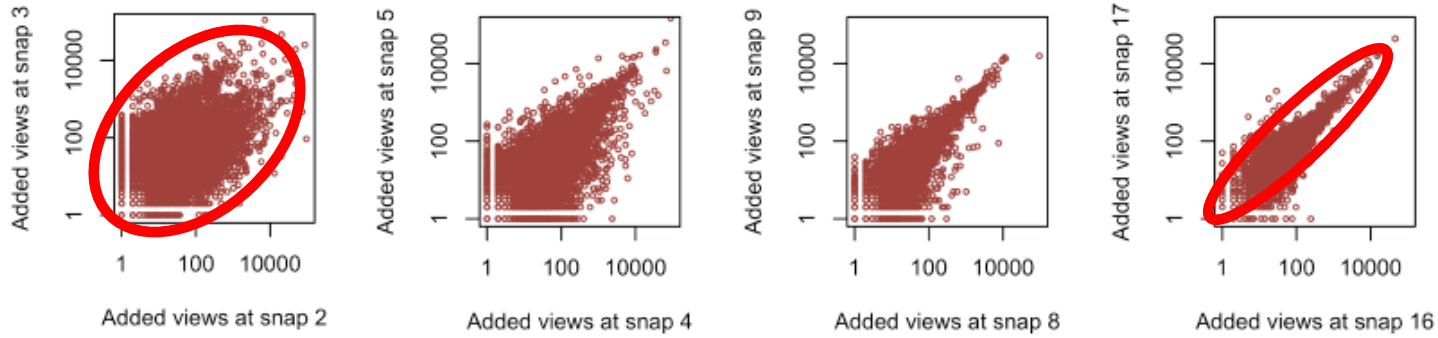
Rich-gets-richer … … and churn



- The more views a video has, the more views it is likely to get in the future
- The relative popularity of the individual videos are highly non-stationary

E.g., IFIP Performance '11

Rich-gets-richer …
… and churn



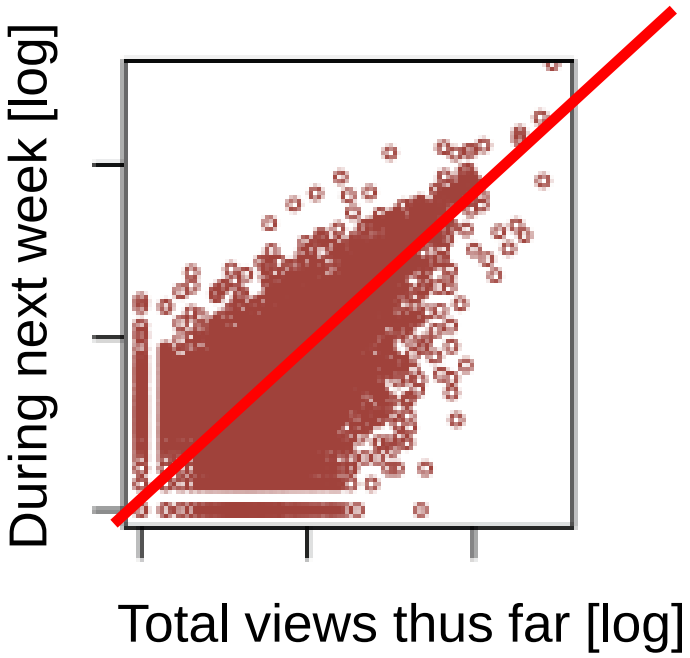
Week 2 Week4 Week 8 Week 16

Young videos Old videos

- The more views a video has, the more views it is likely to get in the future
- The relative popularity of the individual videos are highly non-stationary
- Some long-term popularity

E.g., IFIP Performance '11

Rich-gets-richer …
… and churn



- The more views a video has, the more views it is likely to get in the future
- The relative popularity of the individual videos are highly non-stationary
- Some long-term popularity

E.g., IFIP Performance '11