PageRank

Random Surfers on the Web
Transition Matrix of the Web
Dead Ends and Spider Traps
Topic-Specific PageRank
Hubs and Authorities

Jeffrey D. Ullman
Stanford University/Infolab

Intuition – (1)

- Web pages are important if people visit them a lot.
- But we can't watch everybody using the Web.
- A good surrogate for visiting pages is to assume people follow links randomly.
- Leads to *random surfer/walker* model:
  - Start at a random page and follow random out-links repeatedly, from whatever page you are at.
- *PageRank* of a page = limiting probability of being at that page.

Intuition – (2)

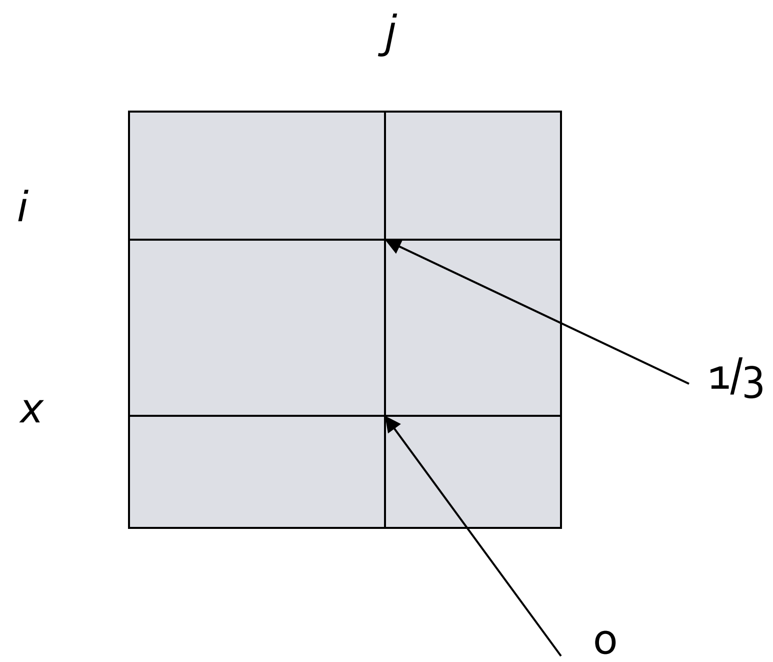
- Solve the recursive equations: "importance of a page = its share of the importance of each of its predecessor pages."
  - Equivalent to the random-surfer definition of PageRank.

Transition Matrix of the Web

- Number the pages 1, 2,… .
  - Page *i* corresponds to row and column *i*.
- $M[i, j] = 1/n$ if page *j* links to *n* pages, including page *i* ; 0 if *j* does not link to *i*.
  - $M[i, j]$ is the probability a surfer will next be at page *i* if it is now at page *j*.
  - Or it is the share of *j*'s importance that *i* receives.
- Note: no matter how many links to page i you find on page j, give i only $1/n^{th}$ of j's importance.

Example: Transition Matrix

Suppose page *j* links to 3 pages, including *i* but not *x.*



Called a (*column*) *stochastic matrix* =
"all columns sum to 1."

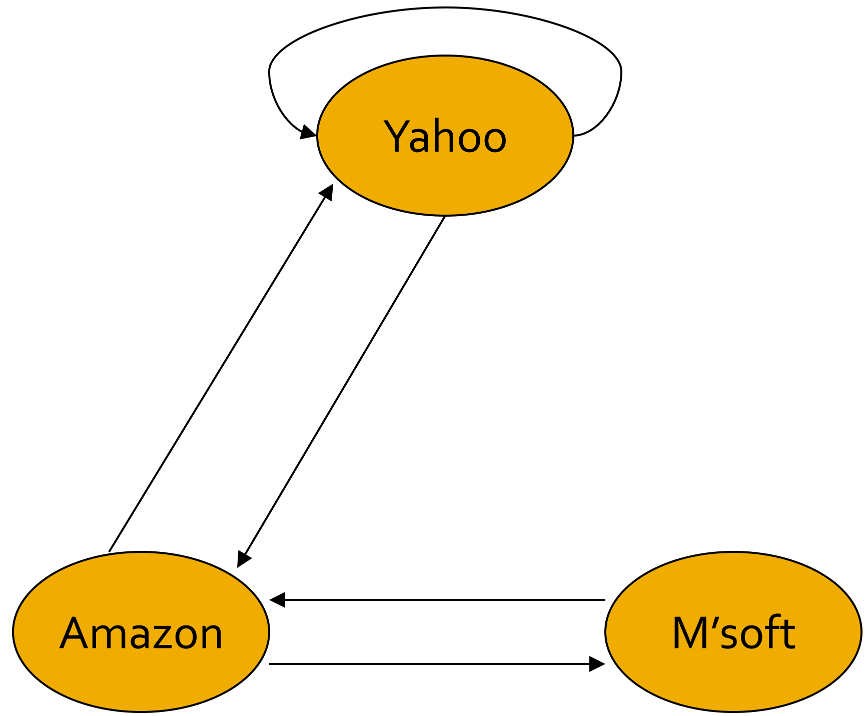
Random Walks on the Web

- Suppose **v** is a vector whose i th component is the probability that a random surfer is at page i at a certain time.
- If a surfer chooses a successor page from page i at random, the probability distribution for surfers is then given by the vector $M\mathbf{v}$.

Random Walks – (2)

- Starting from any vector **u**, the limit *M* (*M* (…*M* (*M* **u**) …)) is the long-term distribution of the surfers.
- The math: limiting distribution = principal eigenvector of *M* = PageRank.
  - Note: If **v** is the limit of MM…M**u**, then **v** satisfies the equation **v** = M**v**, so **v** is an eigenvector of M with eigenvalue 1.

Example: The Web in 1839



$$
\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 1 \\
m & 0 & 1/2 & 0 \\
\hline
\end{array}
$$

Solving The Equations

- Because there are no constant terms, the equations **v** = *M***v** do not have a unique solution.
  - Example: doubling each component of solution **v** yields another solution.
- In Web-sized examples, we cannot solve by Gaussian elimination anyway; we need to use *relaxation* (= iterative solution).

Simulating a Random Walk

- Start with the vector **u** = [1, 1,…, 1] representing the idea that each Web page is given one unit of *importance*.
  - Note: it is more common to start with each vector element = 1/N, where N is the number of Web pages and to keep the sum of the elements at 1.
  - Question for thought: Why such small values?
- Repeatedly apply the matrix *M* to **u**, allowing the importance to flow like a random walk.
- About 50 iterations is sufficient to estimate the limiting solution.

Example: Iterating Equations

- Equations **v** = M**v**:

$y = y/2 + a/2$

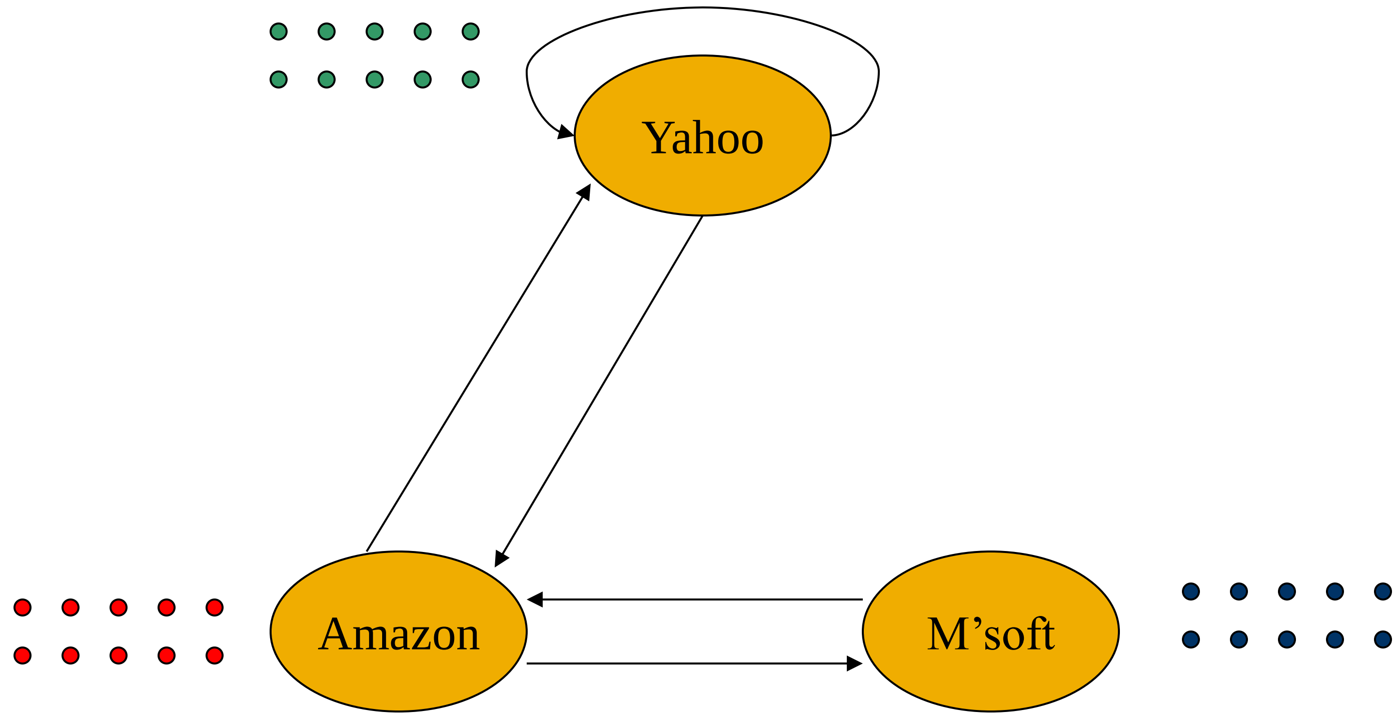
$a = y/2 + m$

$m = a/2$

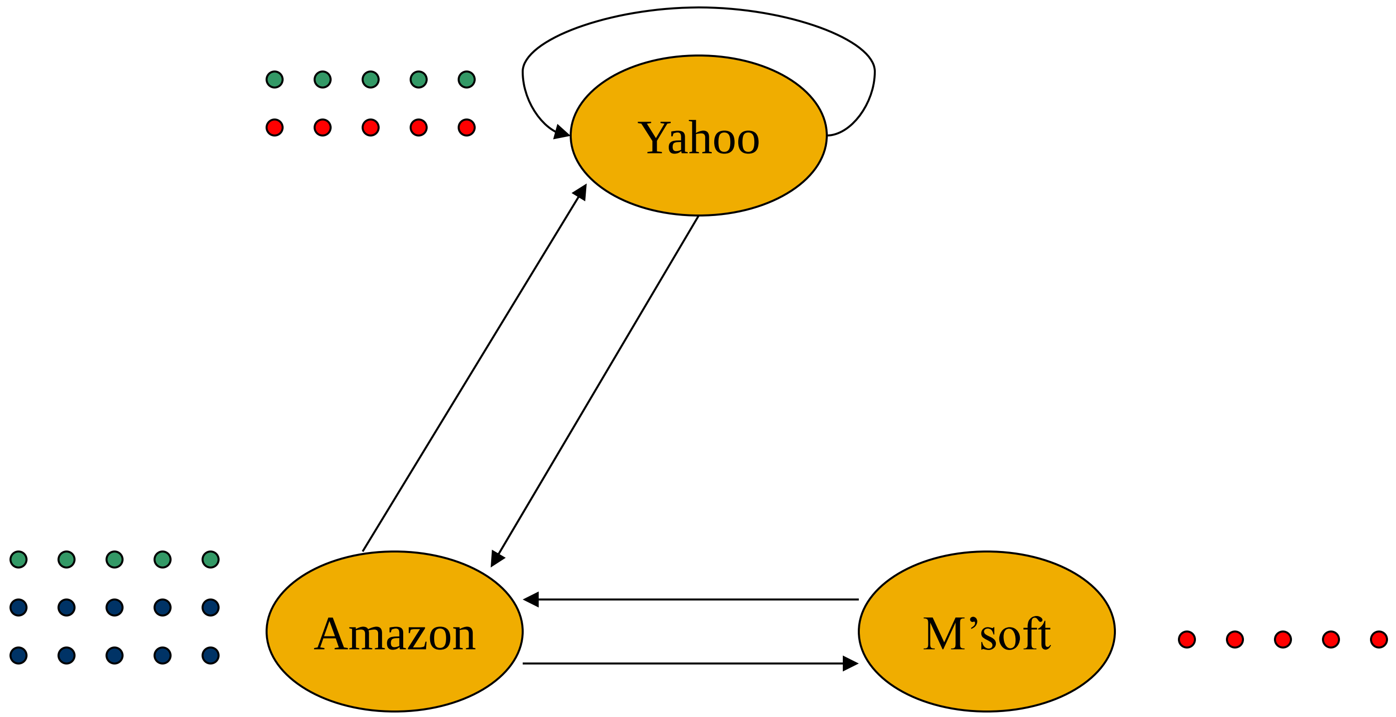
Note: "=" is really "assignment," but in parallel

y		1	1	5/4	9/8		6/5
a	=	1	3/2	1	11/8	. . .	6/5
m		1	1/2	3/4	1/2		3/5

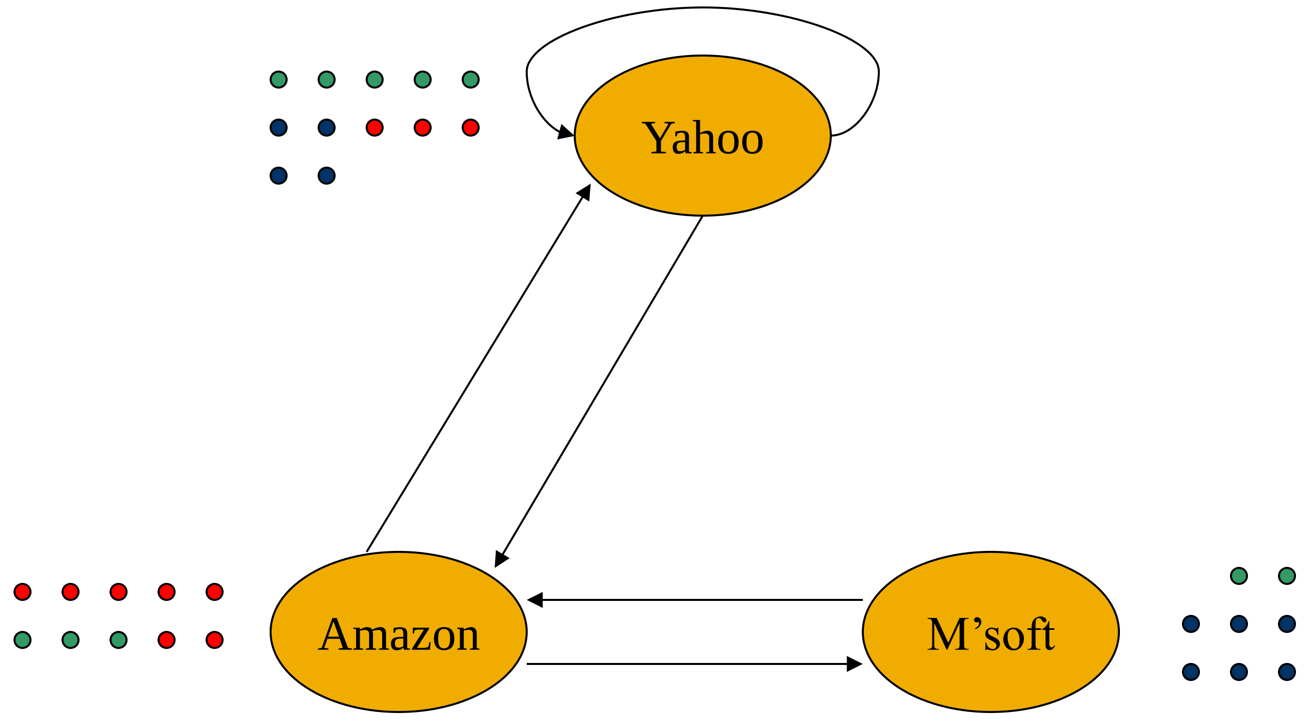
The Surfers



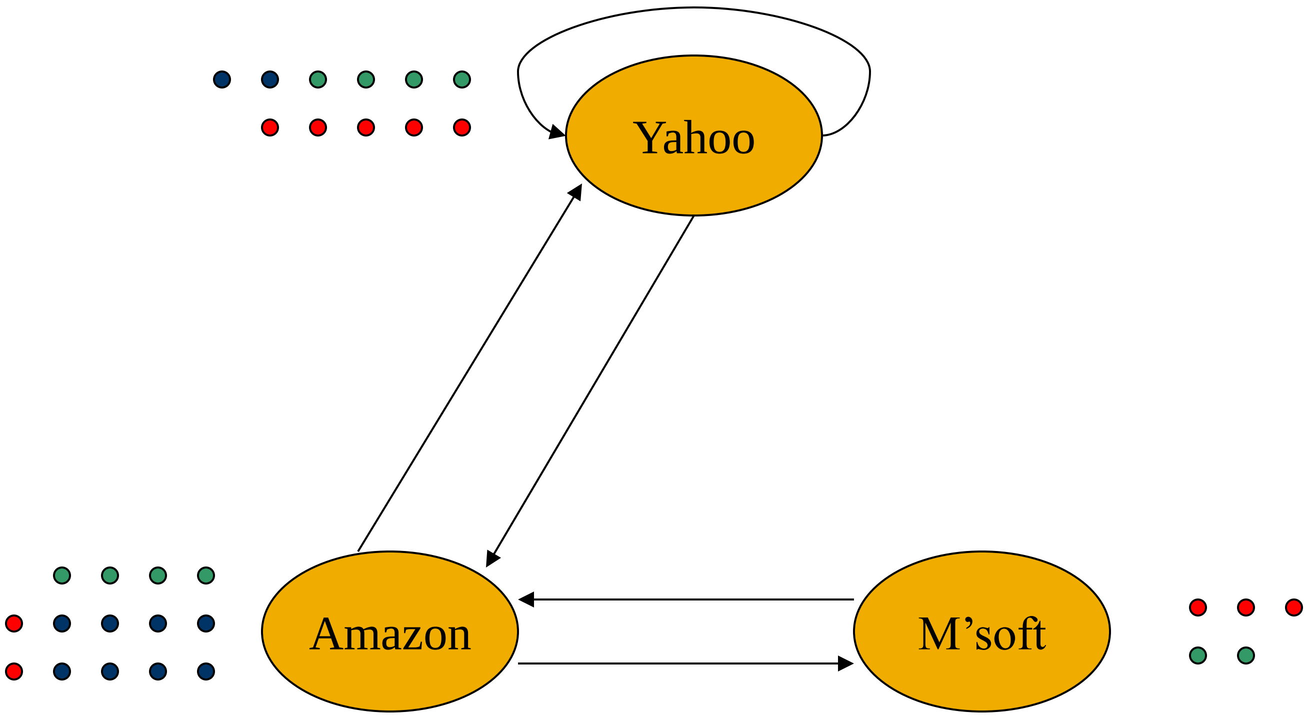
The Surfers



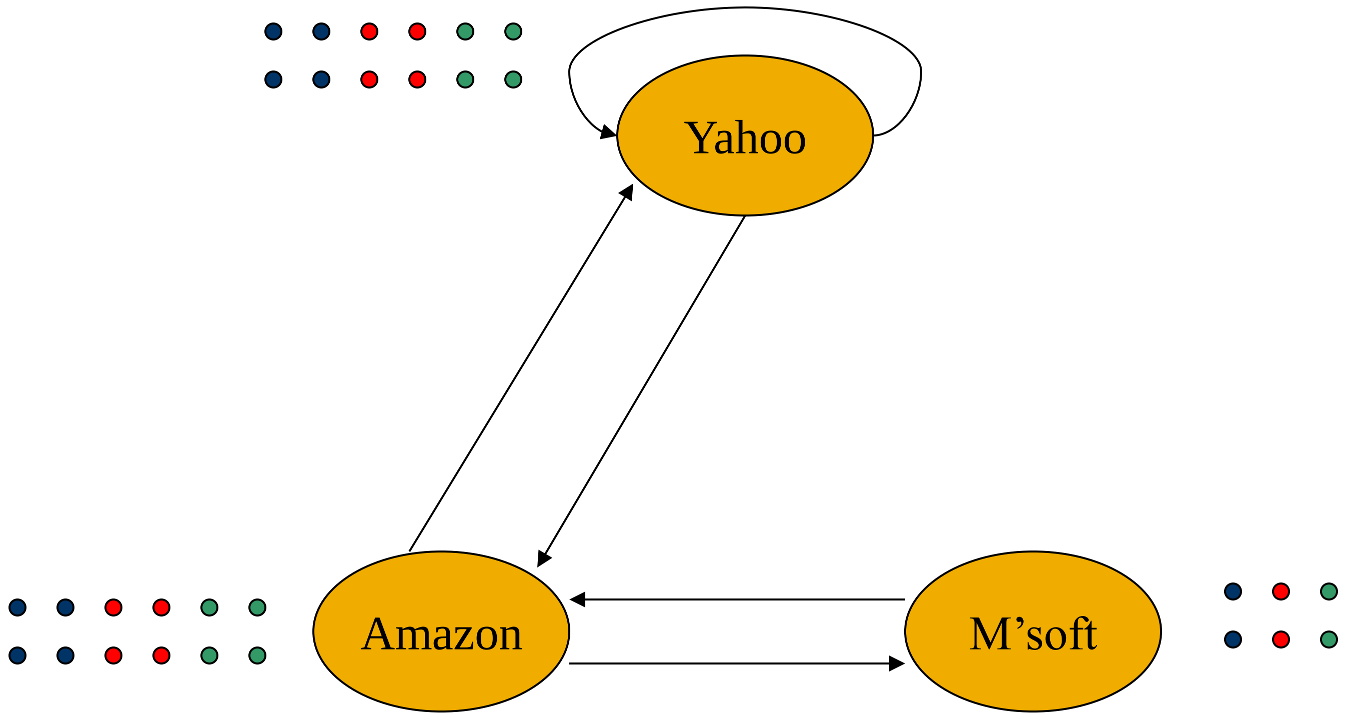
The Surfers



The Surfers



In the Limit …



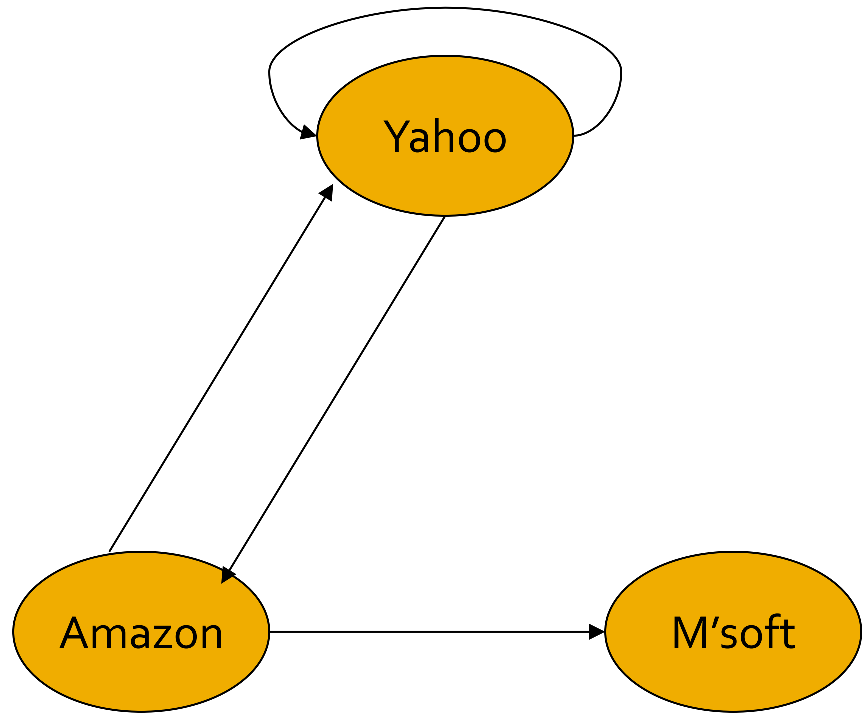
The Web Is More Complex Than That

Dead Ends
Spider Traps
Taxation Policies

Real-World Problems

- Some pages are *dead ends* (have no links out).
  - Such a page causes importance to leak out, or surfers to disappear.
- Other groups of pages are *spider traps* (all out-links are within the group).
  - Eventually spider traps absorb all importance; all surfers get stuck in the trap.

Microsoft Becomes Dead End



$$
\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 0 \\
m & 0 & 1/2 & 0 \\
\hline
\end{array}
$$

A (*column*) *substochastic matrix* =
"all columns sum to at most 1."

Example: Effect of Dead Ends

- Equations **v** = *M***v**:

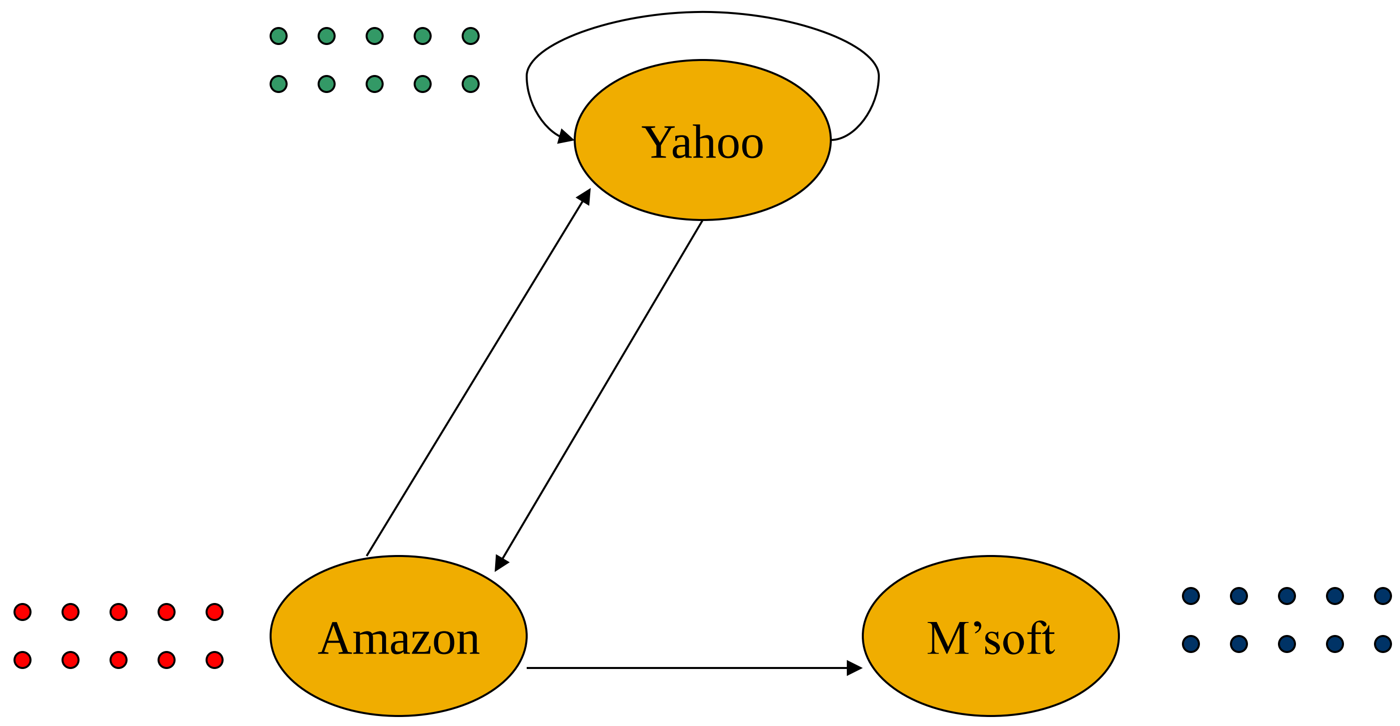
$y = y/2 + a/2$

$a = y/2$

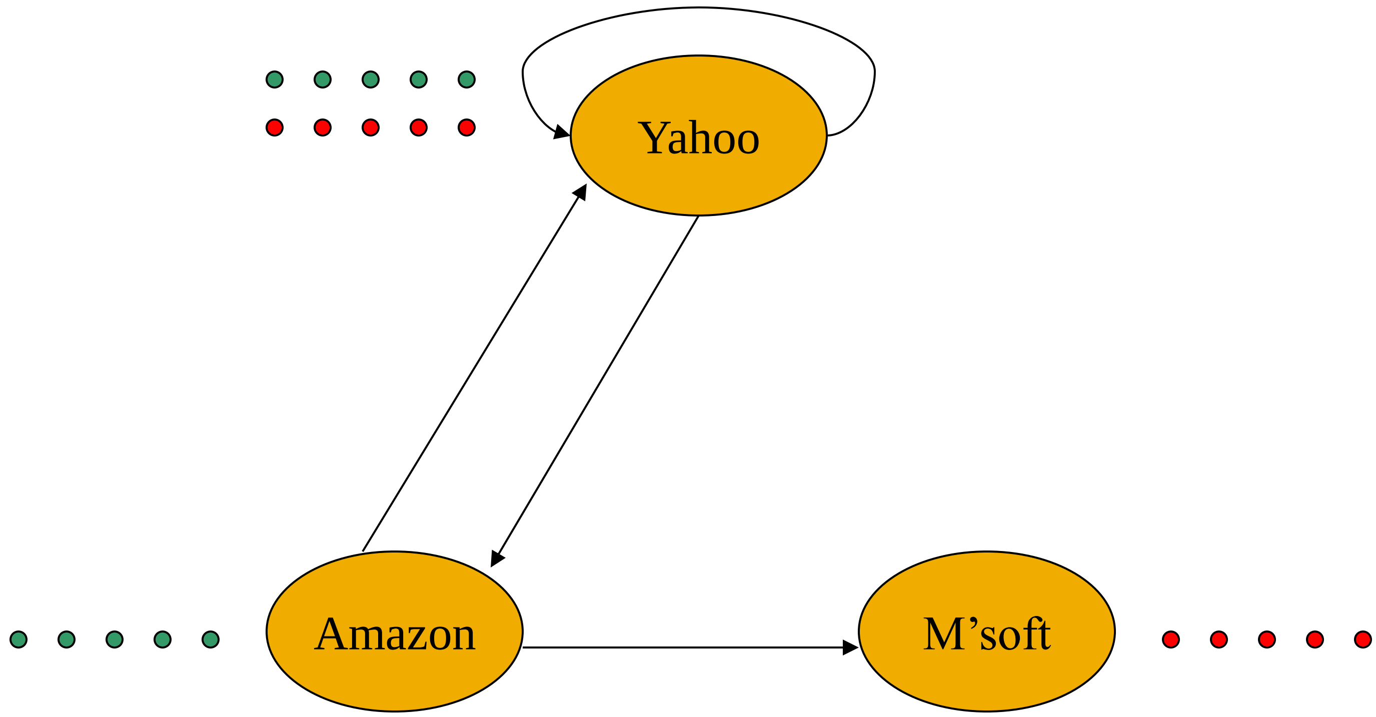
$m = a/2$

y		1	1	3/4	5/8		0
a	=	1	1/2	1/2	3/8	. . .	0
m		1	1/2	1/4	1/4		0

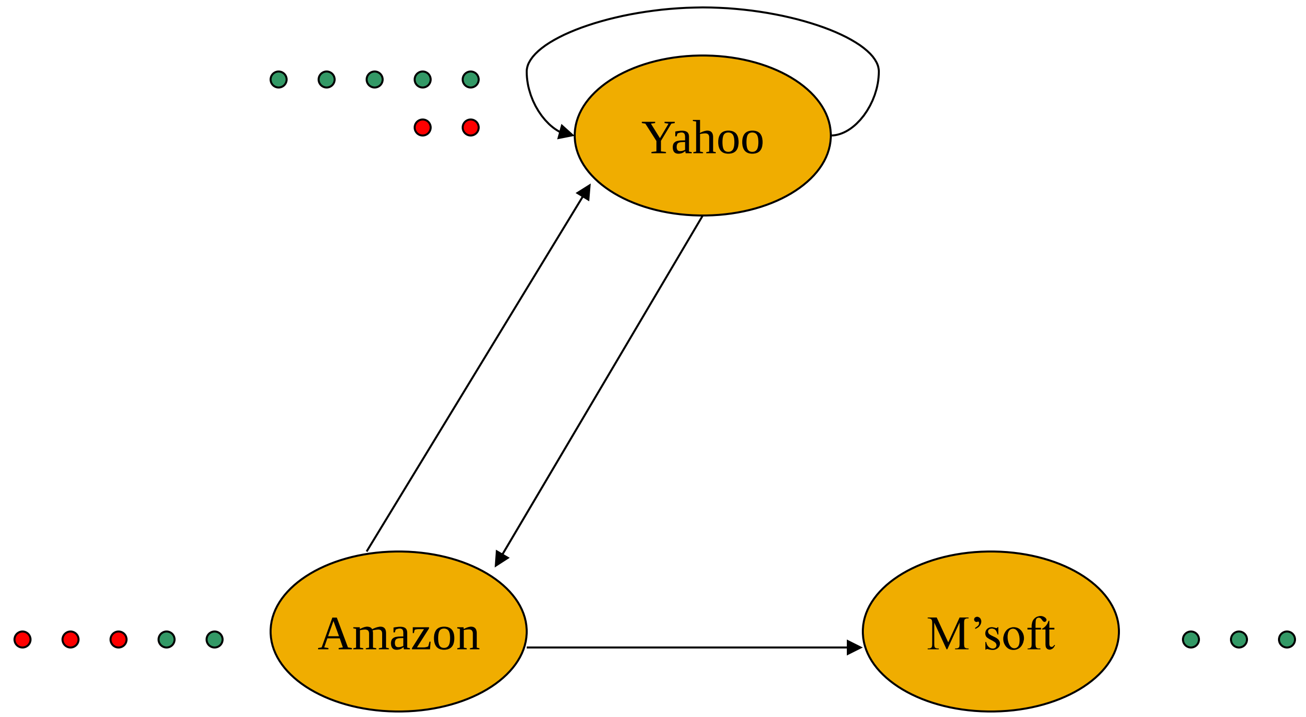
Microsoft Becomes a Dead End



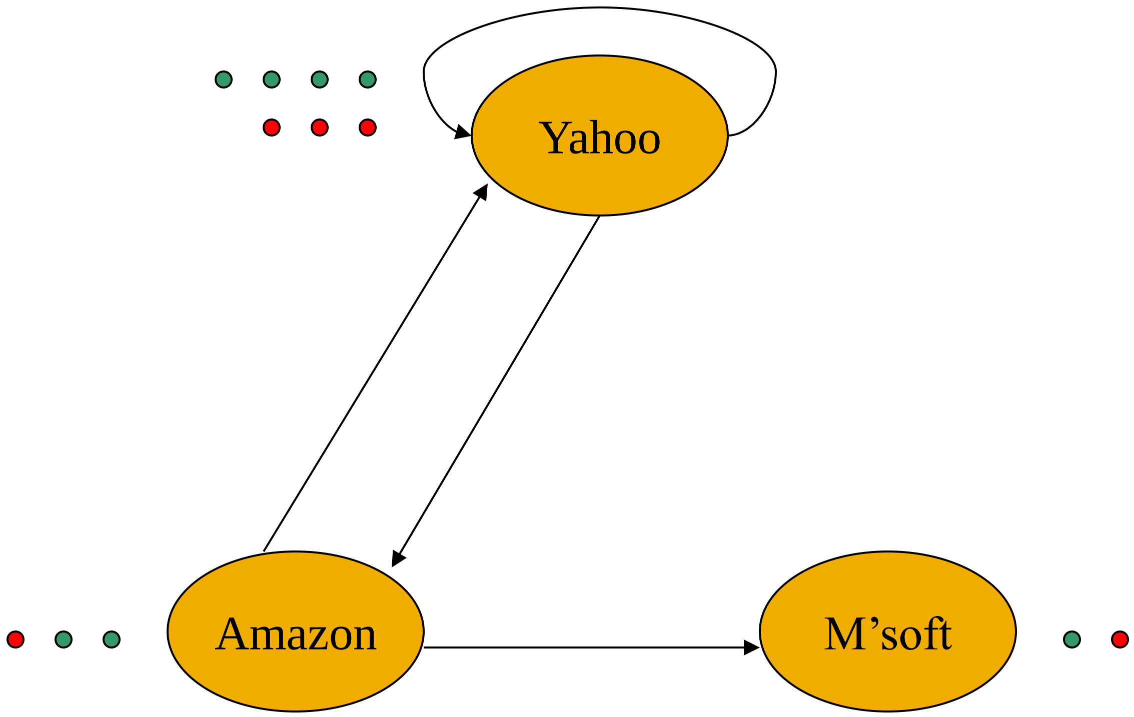
Microsoft Becomes a Dead End



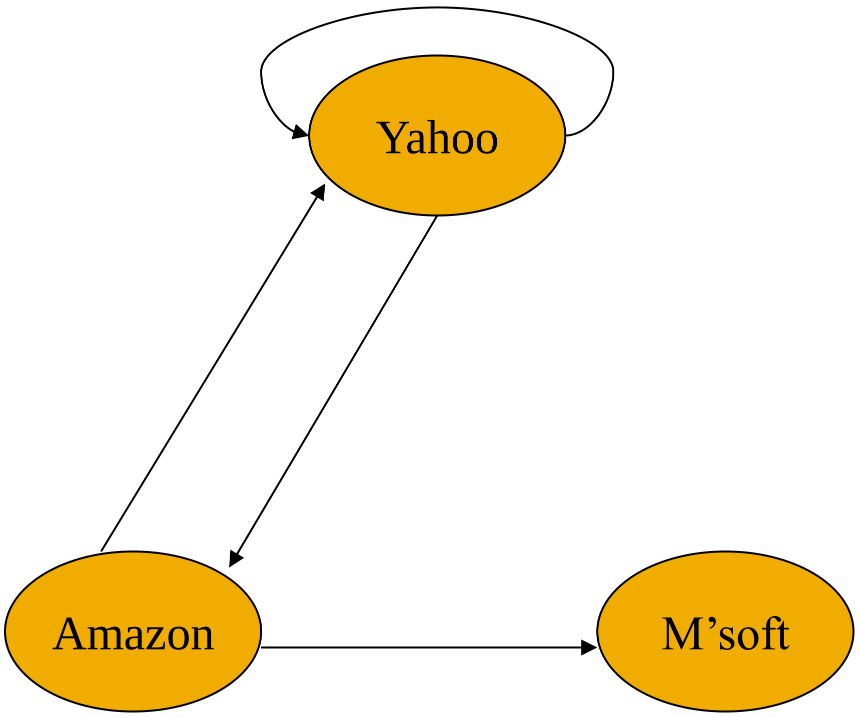
Microsoft Becomes a Dead End



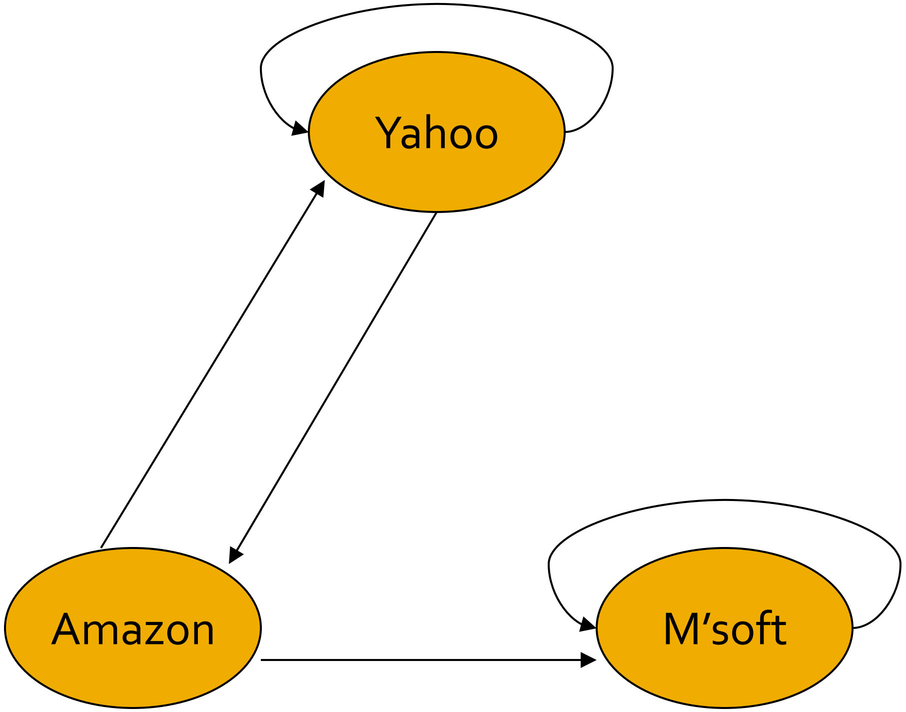
Microsoft Becomes a Dead End



In the Limit ...



M’soft Becomes Spider Trap



$$\begin{array}{c|ccc|} & y & a & m \\ \hline y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 0 \\ m & 0 & 1/2 & 1 \\ \hline \end{array}$$

Example: Effect of Spider Trap

- Equations $\mathbf{v} = M\mathbf{v}$:

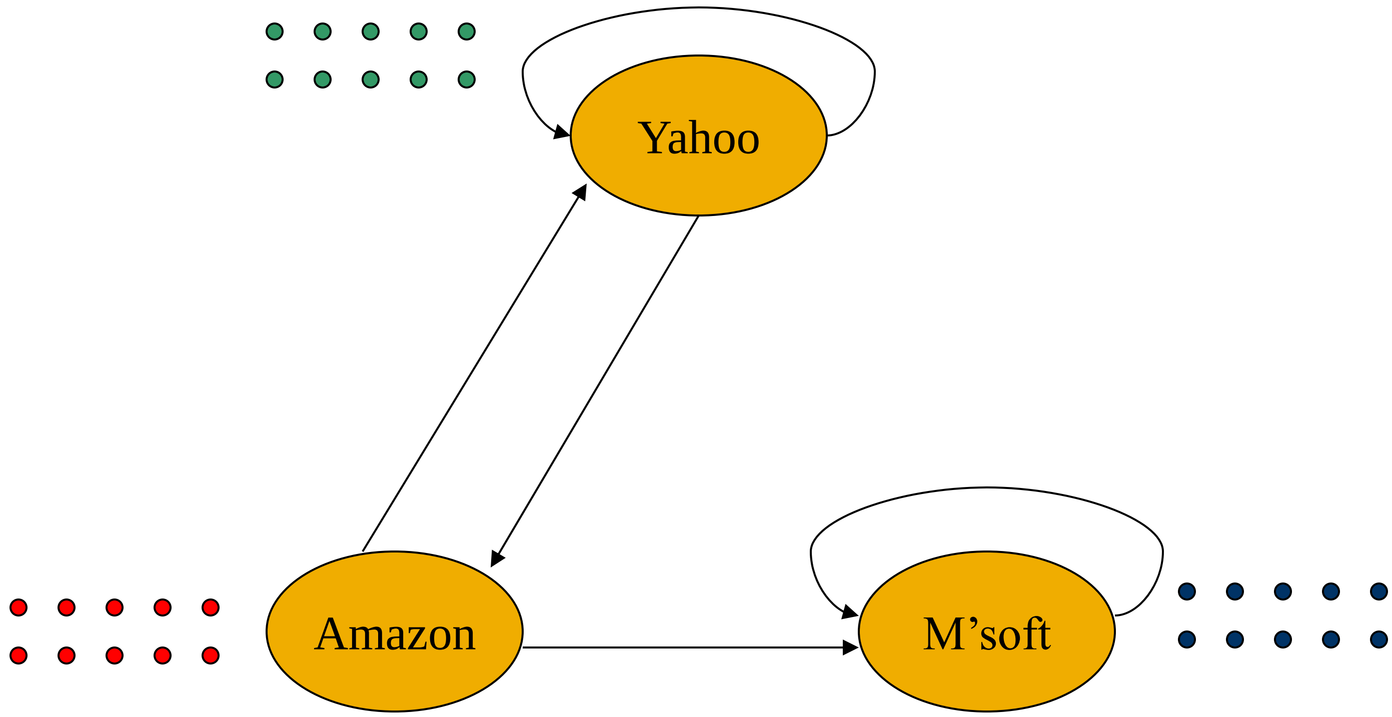
$y = y/2 + a/2$

$a = y/2$

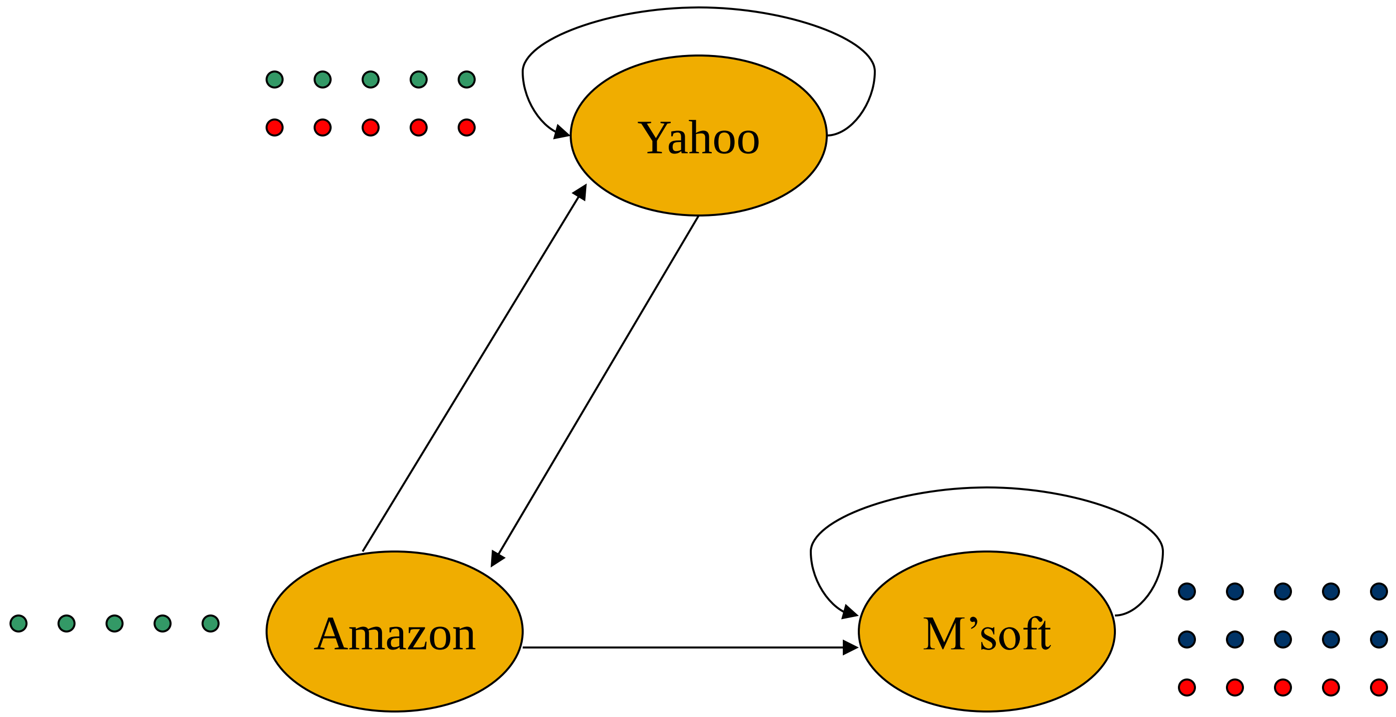
$m = a/2 + m$

$$
\begin{array}{ccccccc}
y & & 1 & 1 & 3/4 & 5/8 & & 0 \\
a & = & 1 & 1/2 & 1/2 & 3/8 & \ldots & 0 \\
m & & 1 & 3/2 & 7/4 & 2 & & 3
\end{array}
$$

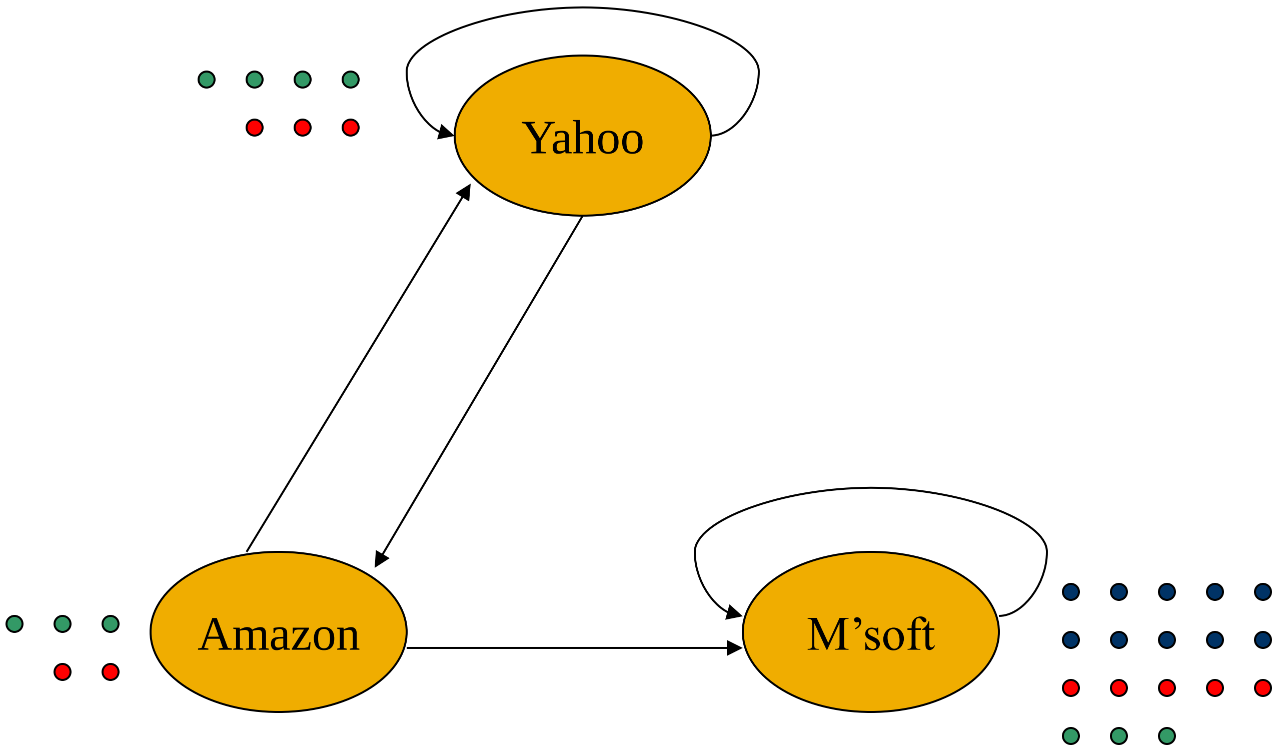
Microsoft Becomes a Spider Trap



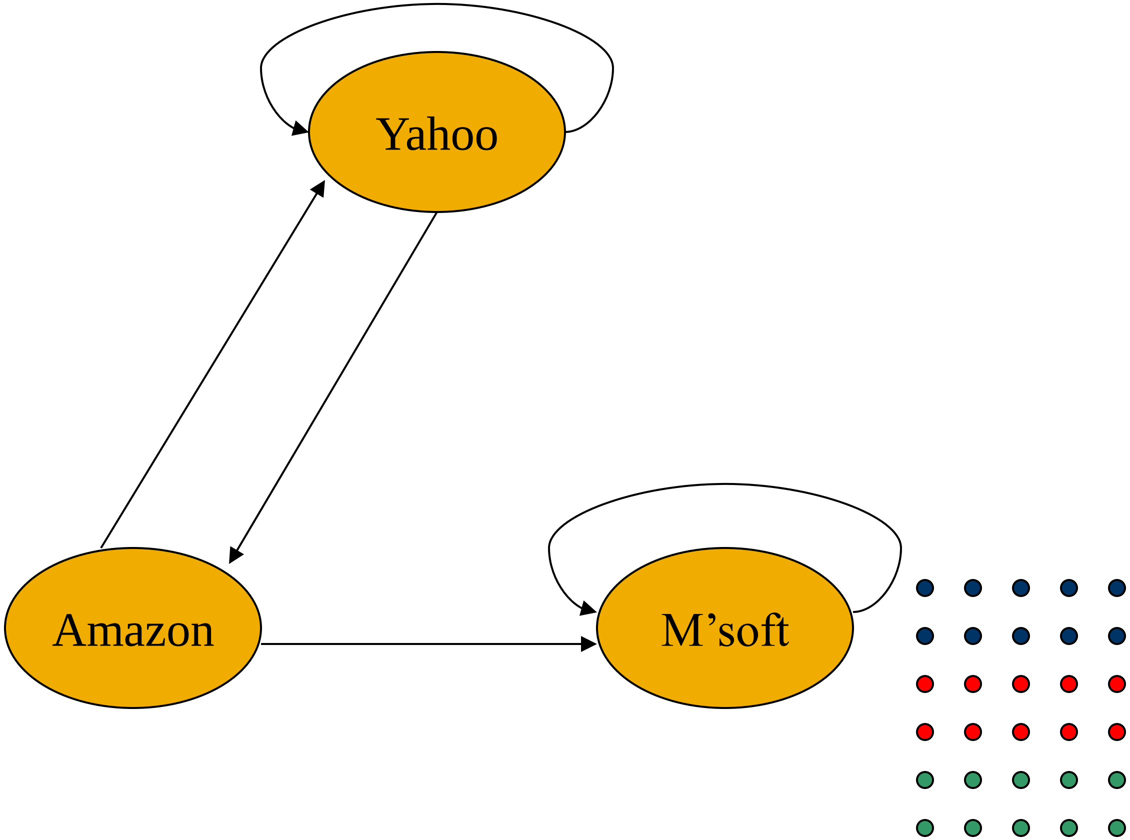
Microsoft Becomes a Spider Trap



Microsoft Becomes a Spider Trap



In the Limit …



PageRank Solution to Traps, Etc.

- "Tax" each page a fixed percentage at each iteration.
- Add a fixed constant > 0 to all pages.
  - Optional but useful: add exactly enough to balance the loss (tax + PageRank of dead ends).
- Models a random walk with a fixed probability of leaving the system, and a fixed number of new surfers injected, at random pages, into the system at each step.

Example: Microsoft is a Spider Trap; 20% Tax

- Equations $\mathbf{v} = 0.8(M\mathbf{v}) + \mathbf{0.2}$:

$$y = 0.8(y/2 + a/2) + 0.2$$

$$a = 0.8(y/2) + 0.2$$

$$m = 0.8(a/2 + m) + 0.2$$

Note: amount injected is chosen to balance the tax. If we started with 1/3 for each rather than 1, the 0.2 would be replaced by 0.0667.

y		1	1.00	0.84	0.776		7/11
a	=	1	0.60	0.60	0.536	. . .	5/11
m		1	1.40	1.56	1.688		21/11

Topic-Specific PageRank

Focusing on Specific Pages
Teleport Sets
Interpretation as a Random Walk

Topic-Specific Page Rank

- Goal: Evaluate Web pages not just by popularity, but also by relevance to a particular topic, e.g. “sports” or “history.”
- Allows search queries to be answered based on interests of the user.
- Example: Search query [jaguar] wants different pages depending on whether you are interested in automobiles, nature, or sports.
  - Might discover interests by recent browsing history, bookmarks, e.g.

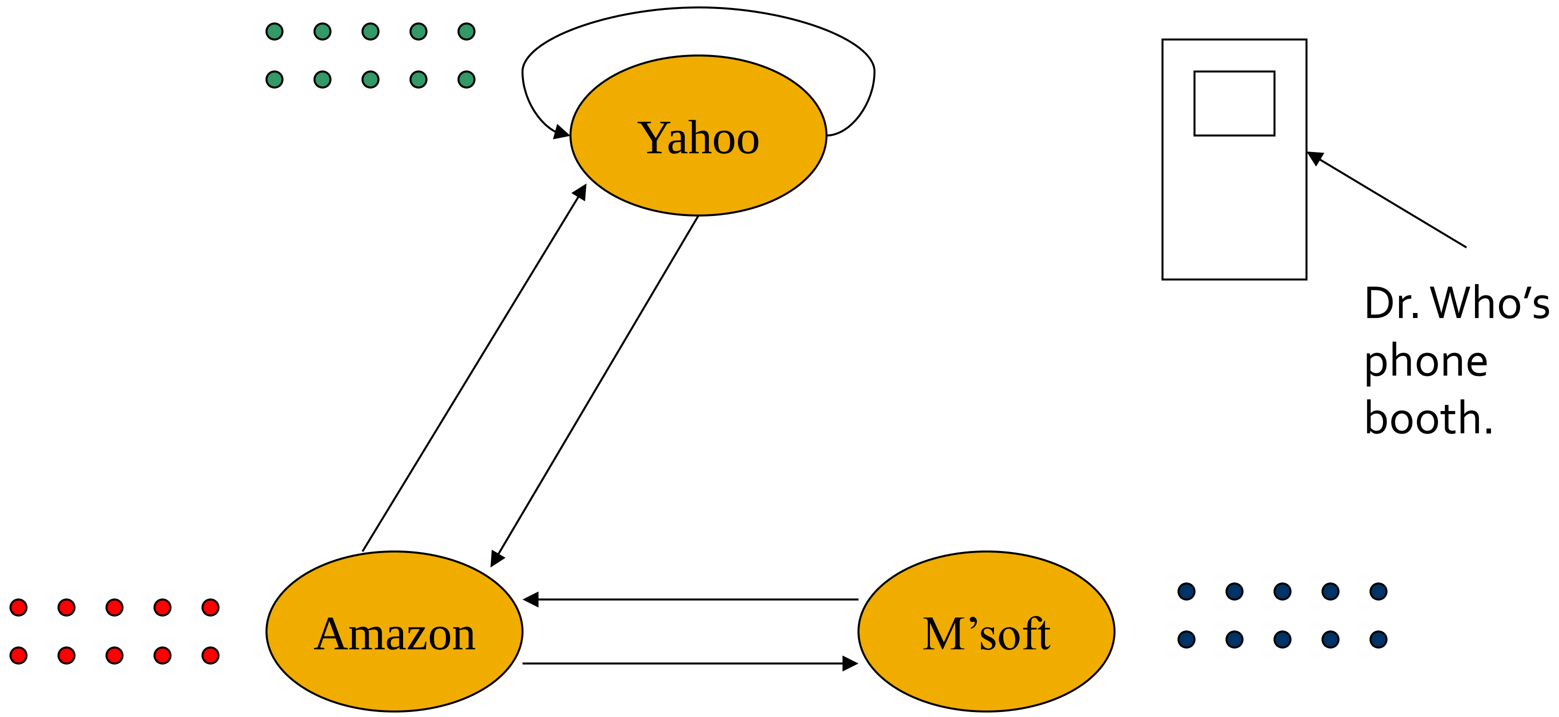
Teleport Sets

- Assume each surfer has a small probability of "teleporting" at any tick.
- Teleport can go to:
  1. Any page with equal probability.
     - As in the "taxation" scheme.
  2. A set of "relevant" pages (*teleport set*).
     - For *topic-specific* PageRank.
- Note: can also inject surfers to compensate for surfers lost at dead ends.
  - Or imagine a surfer always teleports from a dead end.

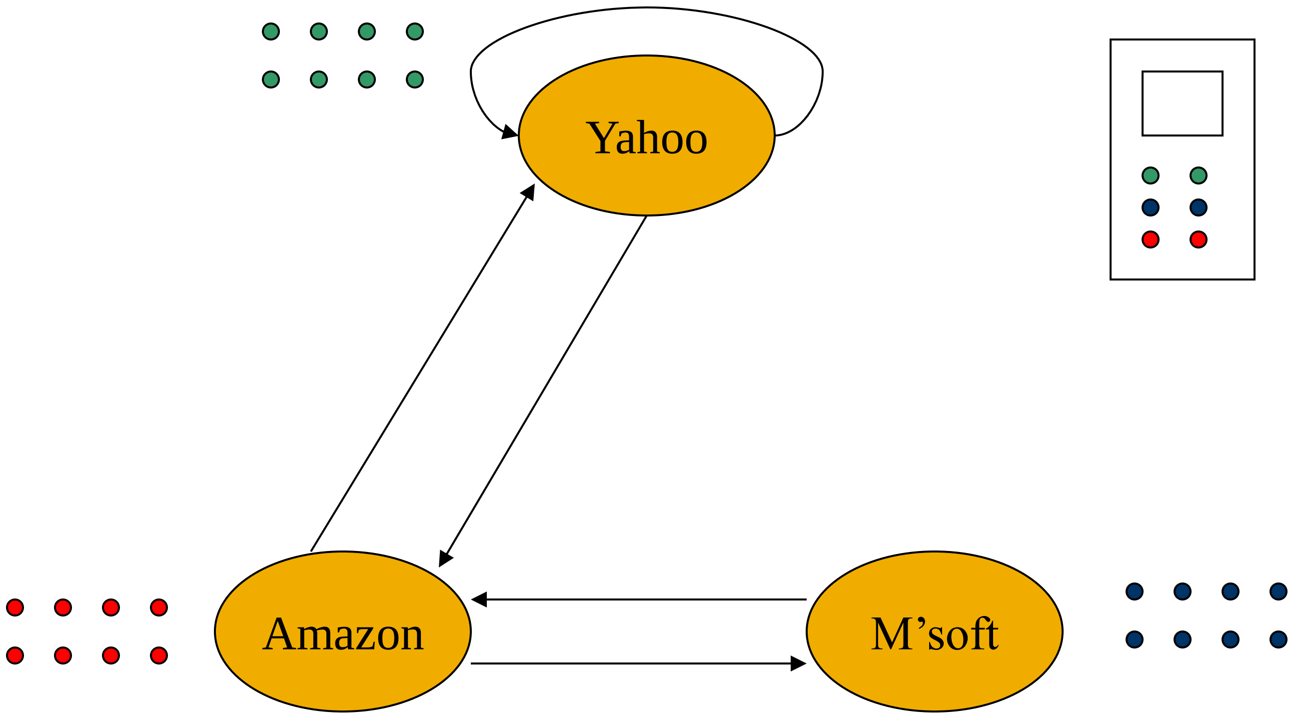
Example: Topic = Software

- Only Microsoft is in the teleport set.
- Assume 20% “tax.”
  - I.e., probability of a teleport is 20%.

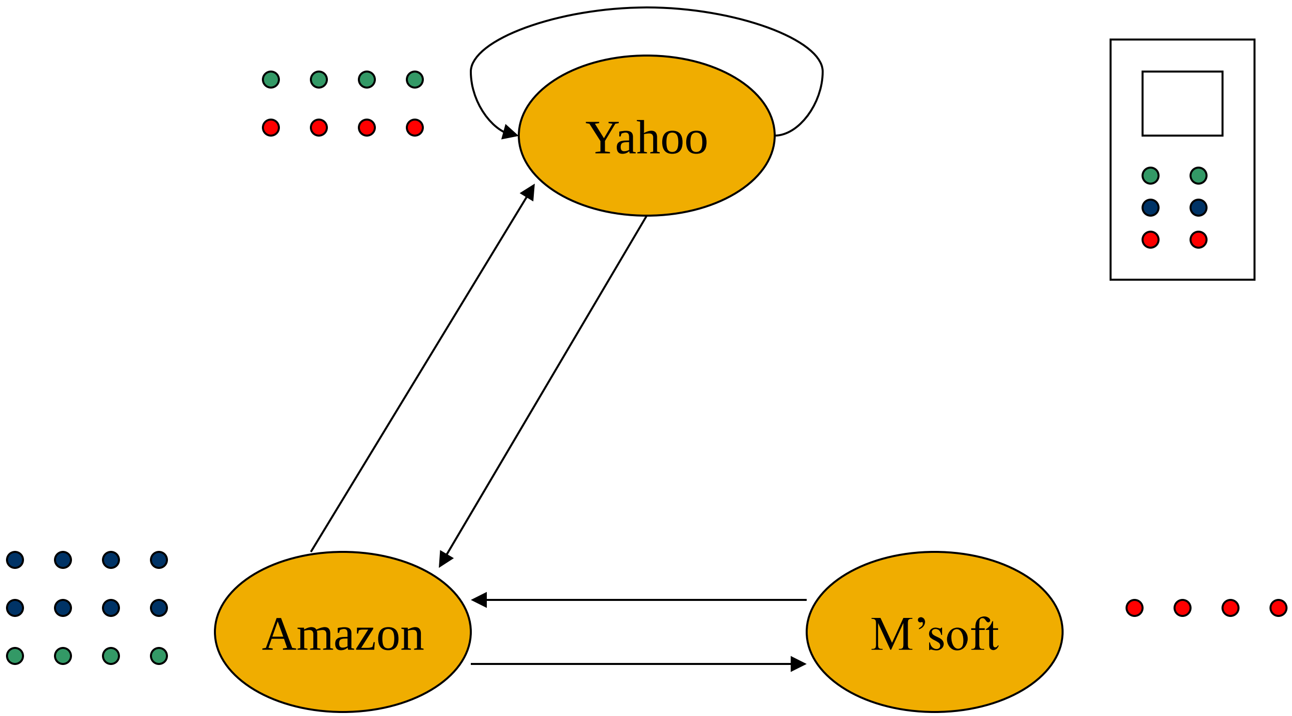
Only Microsoft in Teleport Set



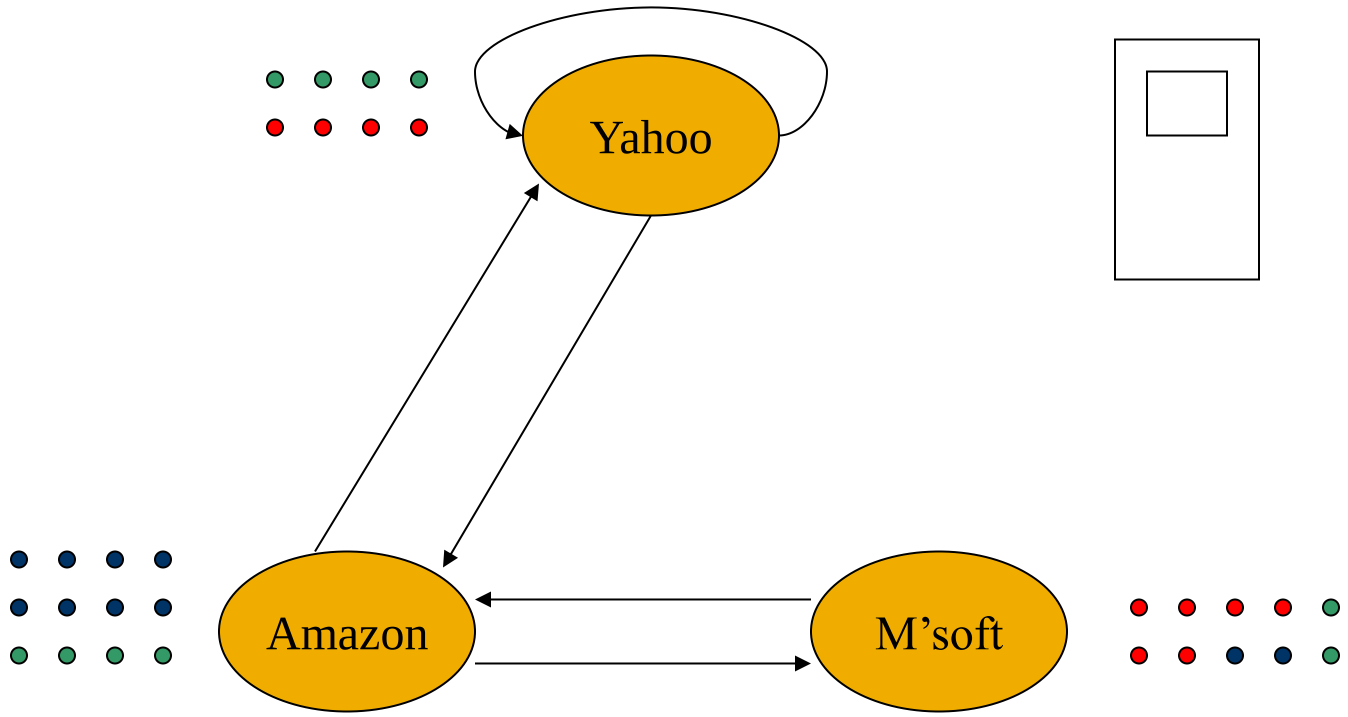
Only Microsoft in Teleport Set



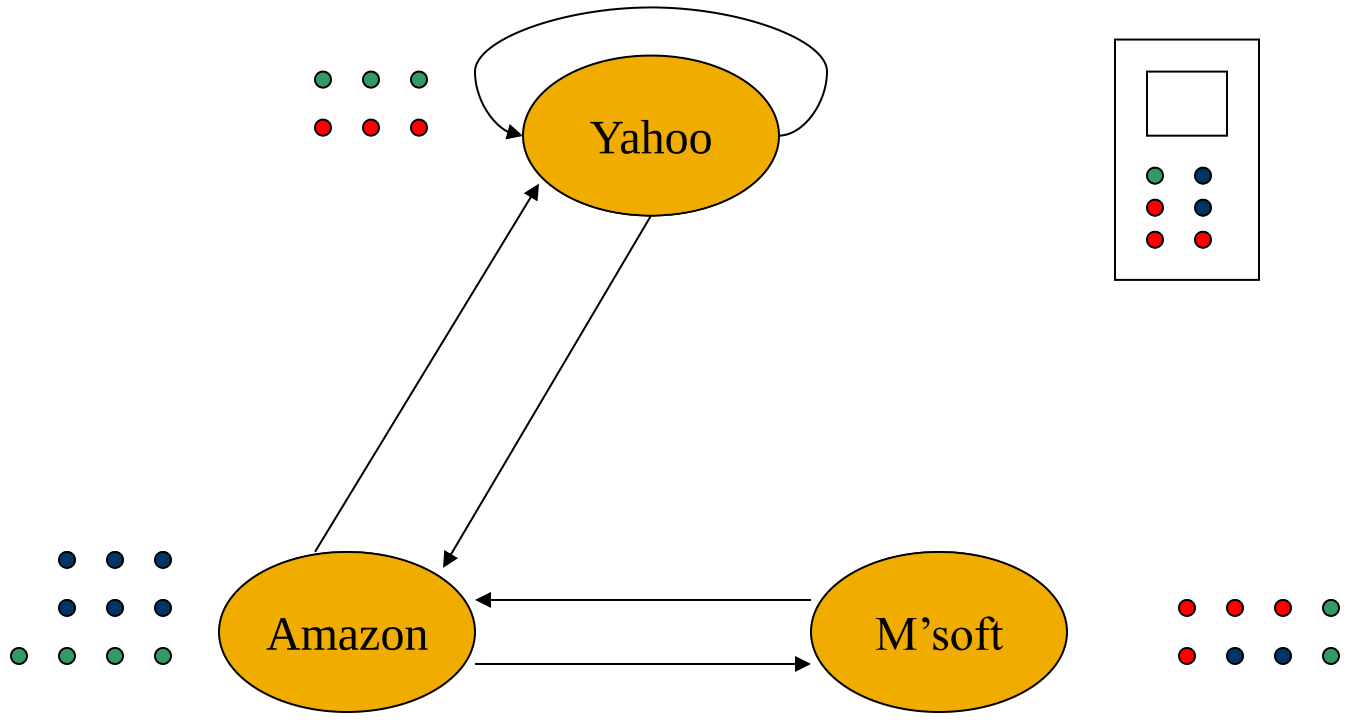
Only Microsoft in Teleport Set



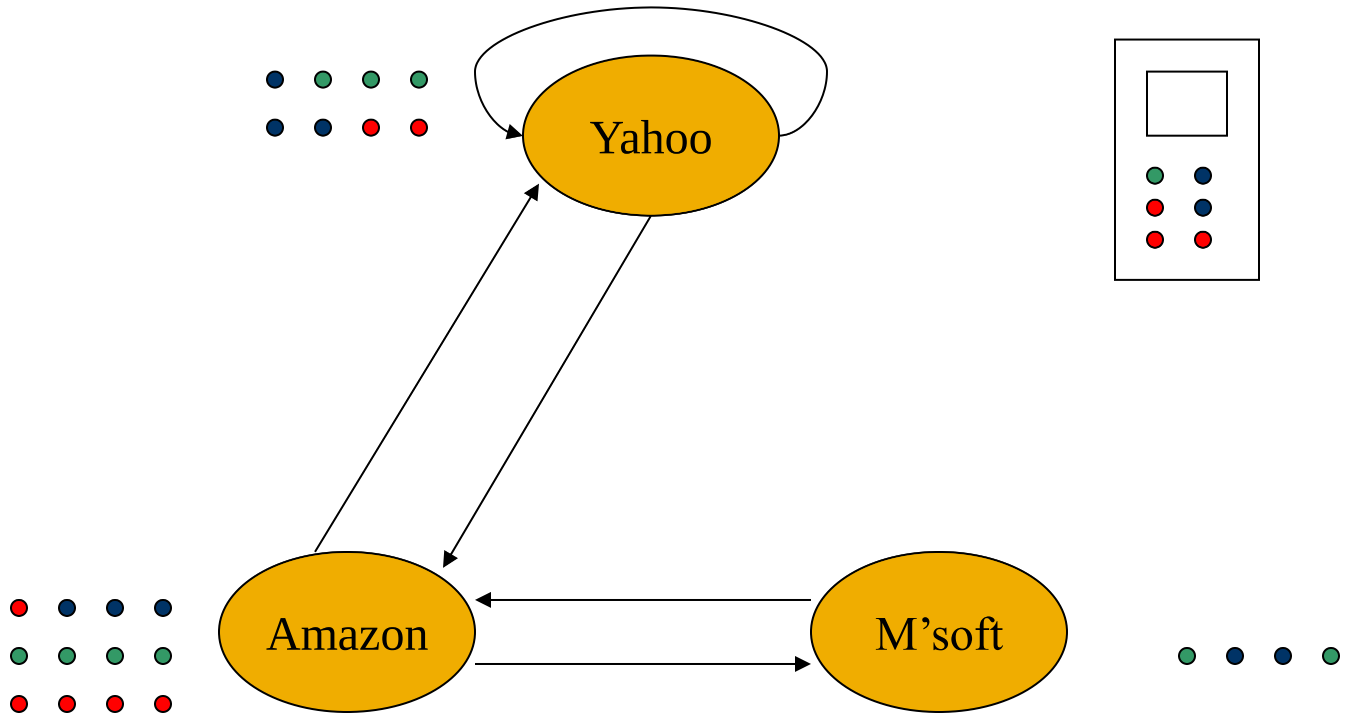
Only Microsoft in Teleport Set



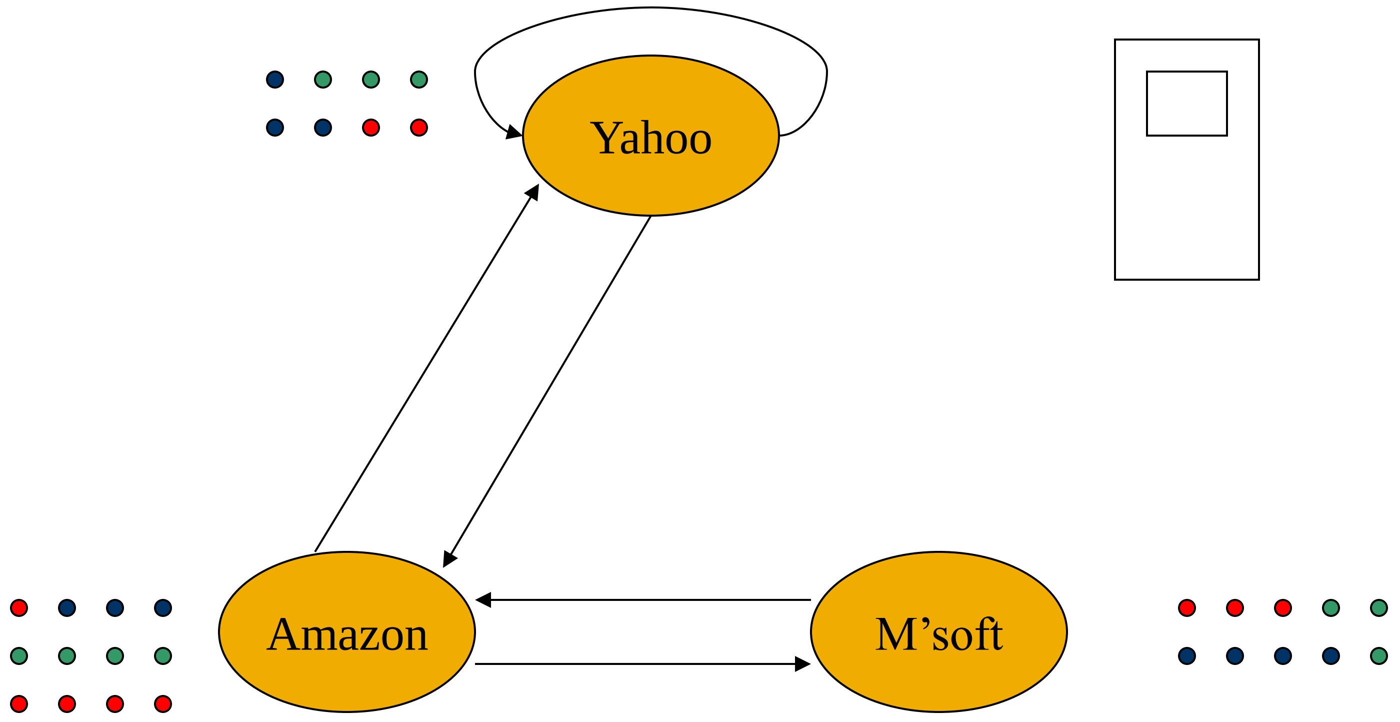
Only Microsoft in Teleport Set



Only Microsoft in Teleport Set



Only Microsoft in Teleport Set



Picking the Teleport Set

1. One option is to choose the pages belonging to the topic in Open Directory.
2. Another option is to “learn,” from a training set (which could be Open Directory), the typical words in pages belonging to the topic; use pages heavy in those words as the teleport set.

Application: Link Spam

- Spam farmers create networks of millions of pages designed to focus PageRank on a few undeserving pages.
  - We’ll discuss this “technology” next time.
- To minimize their influence, use a teleport set consisting of trusted pages only.
  - Example: home pages of universities.

HITS

Hubs
Authorities
Solving the Implied Recursion

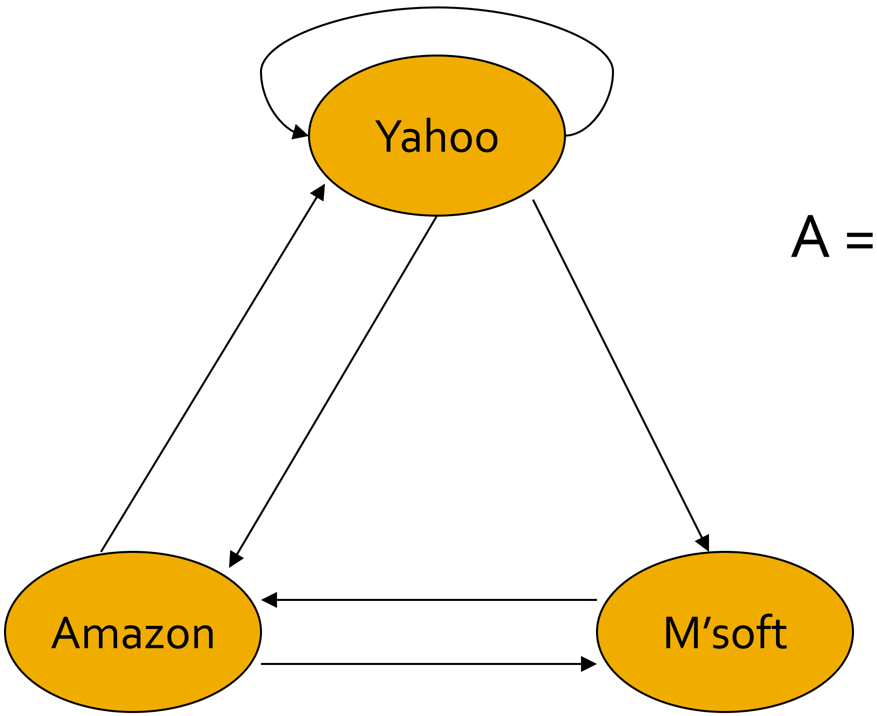
Hubs and Authorities ("HITS")

- Mutually recursive definition:
  - A *hub* links to many authorities;
  - An *authority* is linked to by many hubs.
- Authorities turn out to be places where information can be found.
  - Example: course home pages.
- Hubs tell where the authorities are.
  - Example: departmental course-listing page.

Transition Matrix A

- HITS uses a matrix $A[i, j] = 1$ if page i links to page j, 0 if not.
- A^T, the transpose of A, is similar to the PageRank matrix M, but A^T has 1's where M has fractions.
- Also, HITS uses column vectors **h** and **a** representing the degrees to which each page is a hub or authority, respectively.
- Computation of **h** and **a** is similar to the iterative way we compute PageRank.

Example: H&A Transition Matrix



$$A = \begin{array}{cc} & \begin{array}{ccc} y & a & m \end{array} \\ \begin{array}{c} y \\ a \\ m \end{array} & \boxed{\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{array}} \end{array}$$

Using Matrix A for HITS

- Powers of A and A^T have elements whose values grow exponentially, so we need scale factors λ and μ.
- Let **h** and **a** be column vectors measuring the "hubbiness" and authority of each page.
- Equations: $\mathbf{h} = \lambda A\mathbf{a}$; $\mathbf{a} = \mu A^T\mathbf{h}$.
  - Hubbiness = scaled sum of authorities of successor pages (out-links).
  - Authority = scaled sum of hubbiness of predecessor pages (in-links).

Consequences of Basic Equations

- From $\mathbf{h} = \lambda A\mathbf{a}$; $\mathbf{a} = \mu A^T\mathbf{h}$ we can derive:
  - $\mathbf{h} = \lambda\mu AA^T\mathbf{h}$
  - $\mathbf{a} = \lambda\mu A^TA\,\mathbf{a}$
- We could compute **h** and **a** by iteration, assuming initially each page has one unit of hubbiness and one unit of authority.
- Technically, these equations let you solve for $\lambda\mu$ as well as **h** and **a**.
- In practice, you don't fix $\lambda\mu$, but rather scale the result at each iteration.
  - Example: scale to keep largest value at 1.

Scale Doesn't Matter

- Remember: it is only the direction of the vectors, or the relative hubbiness and authority of Web pages that matters.
- As for PageRank, the only reason to worry about scale is so you don't get overflows or underflows in the values as you iterate.

Example: Iterating H&A

$$\mathbf{a} = \lambda\mu A^T A\mathbf{a};\ \mathbf{h} = \lambda\mu A A^T \mathbf{h}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad AA^T = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad A^TA = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

a(yahoo)	=	1	5	24	114	...	$1+\sqrt{3}$
a(amazon)	=	1	4	18	84	...	2
a(microsoft)	=	1	5	24	114	...	$1+\sqrt{3}$

h(yahoo)	=	1	6	28	132	...	1.000
h(amazon)	=	1	4	20	96	...	0.735
h(microsoft)	=	1	2	8	36	...	0.268

A Better Way to Solve HITS

- Start with **h** = [1,1,…,1]; multiply by A^T to get first **a**; scale so largest component = 1; then multiply by A to get next **h**, and repeat until approximate convergence.
- You may be tempted to compute AA^T and A^TA first, then iterate multiplication by these matrices, as for PageRank.
- Question for thought: Why is the separate calculations of **h** and **a** actually less efficient than the method suggested on this slide?