

Technical changes since the last Tor talk

Nick Mathewson
The Tor Project
<nickm@torproject.org>

Marty!

We've got to go back to ~~the future~~2004!

- Tor was working, usable, and seemed pretty secure. (v 0.0.7.2)
- Pretty small network.
- No GUI—hard to use.
- We got a couple of Defcon talks!

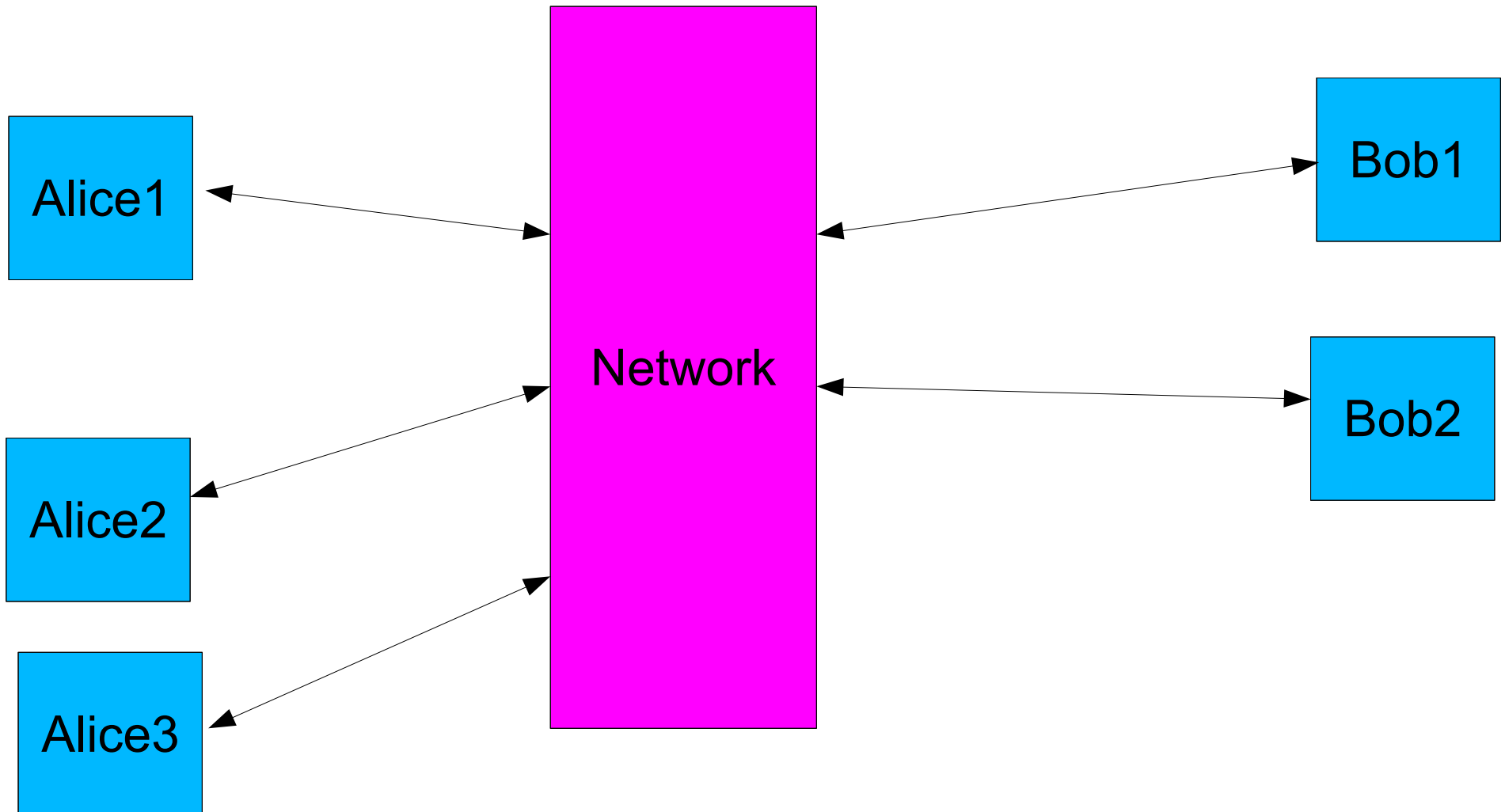
What we've been up to since then.

- Hacking on Tor. (Latest is 0.2.0.4-alpha)
 - Security: adding features / fixing security bugs.
 - Scalability: adding capacity is hard.
 - Scalability: using capacity is hard.
 - Usability: adding GUIs, fixing bugs.
 - Integration: working nice with other apps is hard.
 - Lots more: See the changelog.
- Growing the network: ~200kuser, ~1kserver.

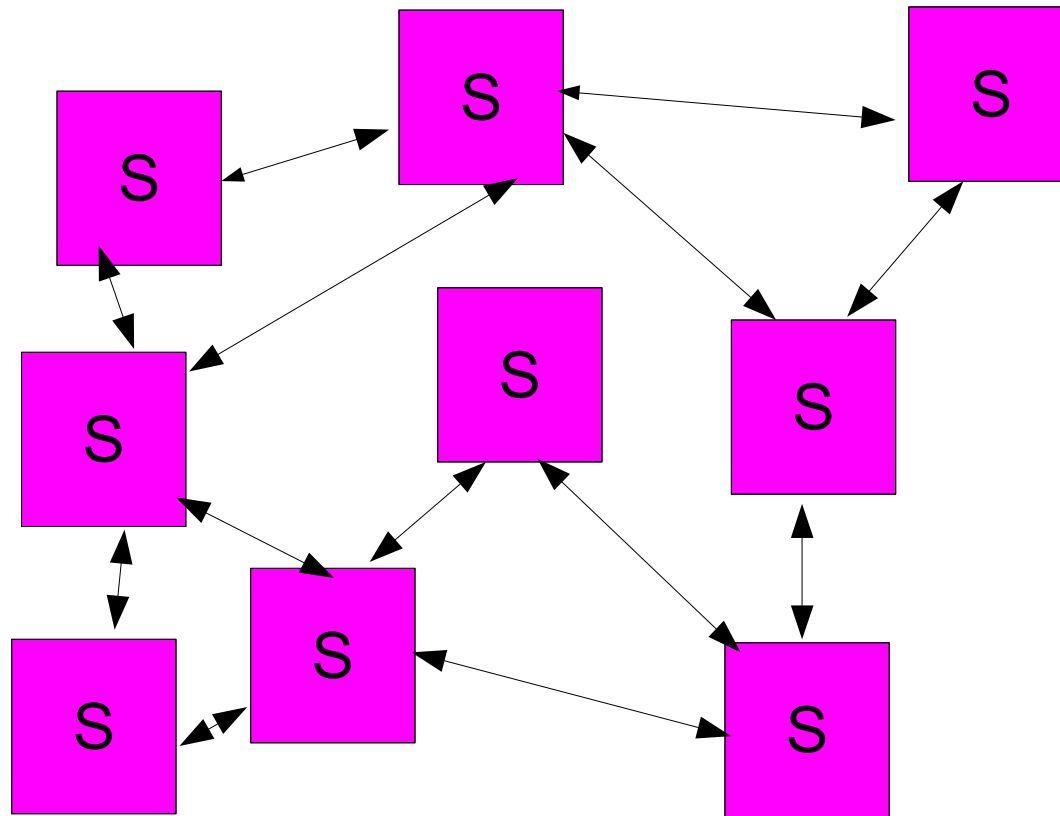
Outline

- Prelude: brief, fast introduction to Tor
- Directories and server discovery changes:
More secure, more scalable!
- Path generation changes:
More efficient, less filling!
- Circuit-building protocol changes:
Oops. Crypto is hard.
- Some fun new tools and features:
What do you mean, I need to edit a file?

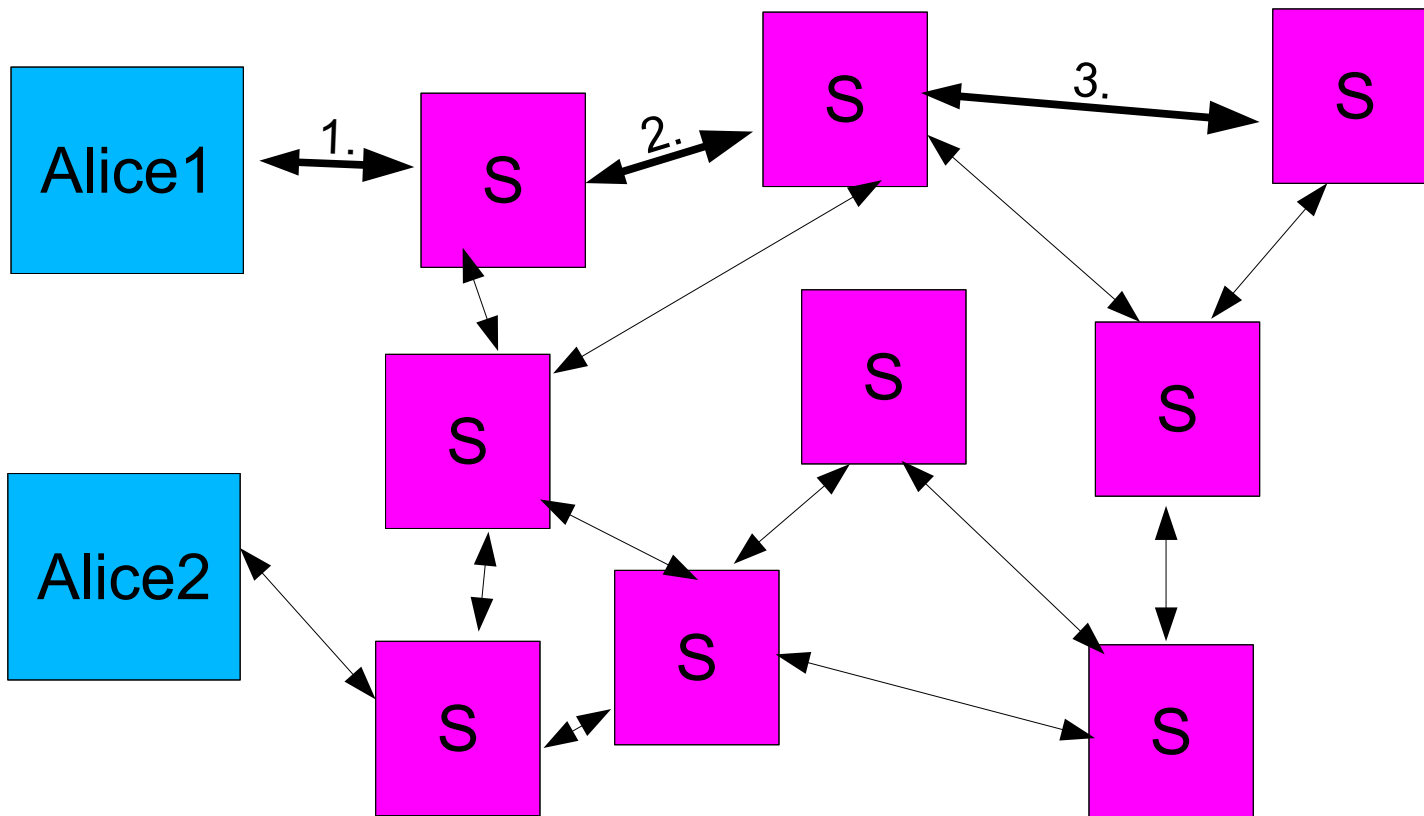
Intro anonymity: anonymity networks hide users among users.



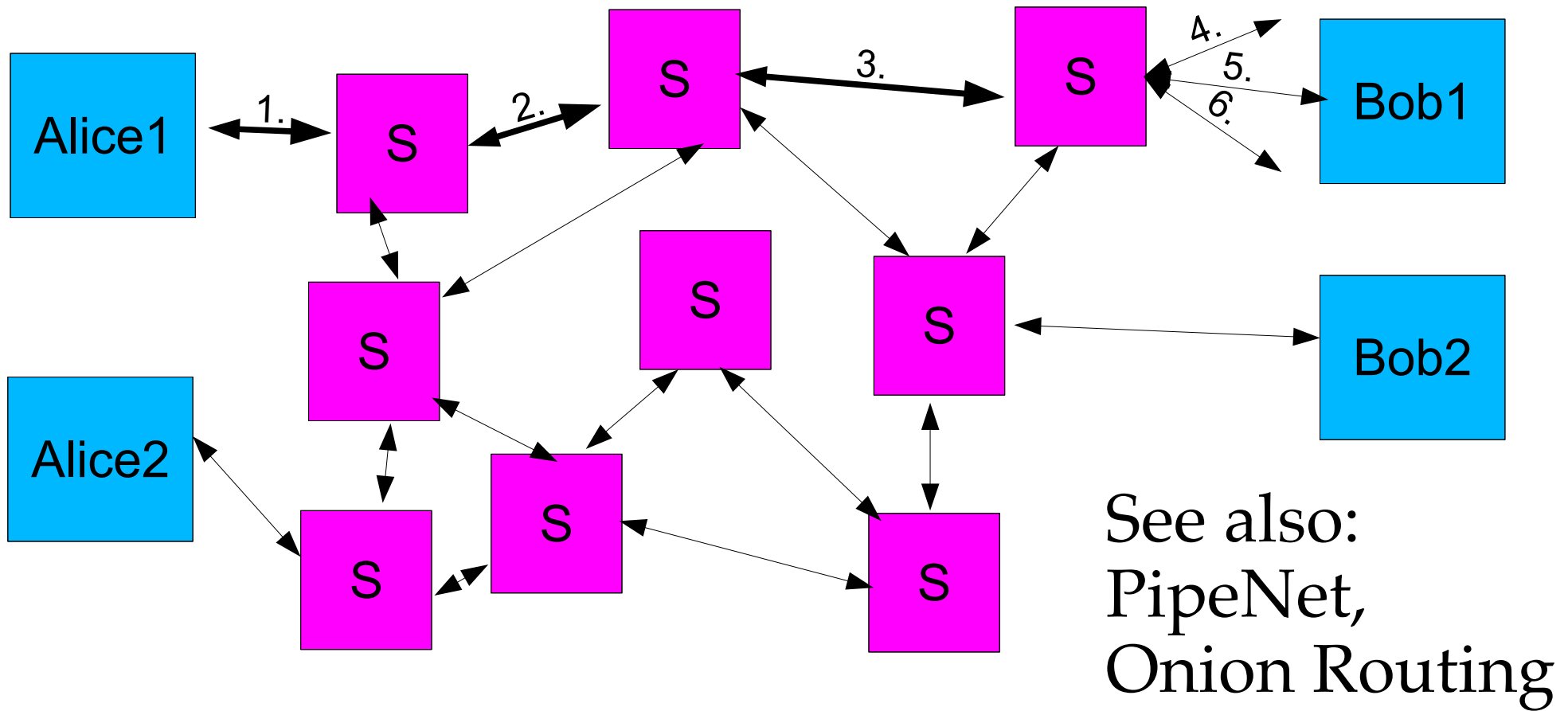
**Intro Tor: There are a bunch of servers,
connected via TLS (ssl).**



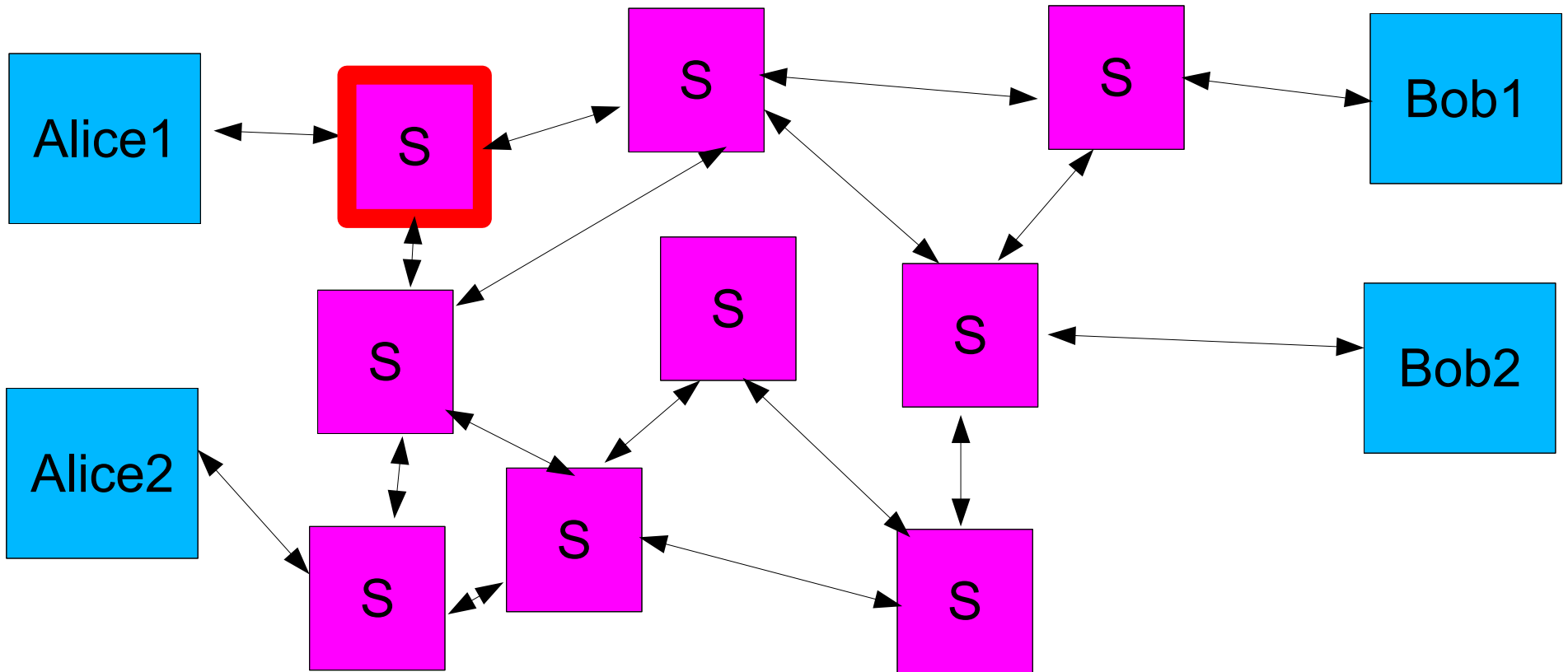
Intro Tor: clients build circuits through a network of decrypting relays.



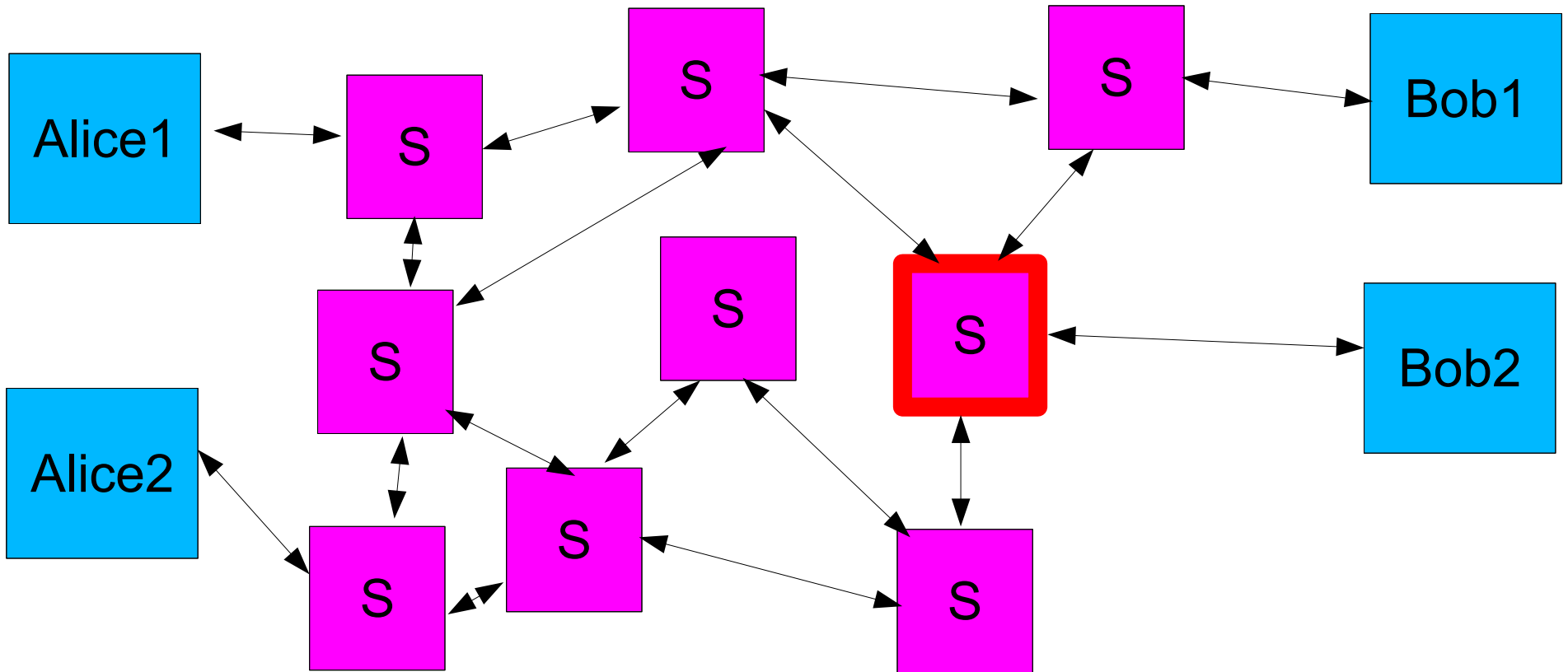
Intro Tor: circuits are used to relay multiple TCP streams.



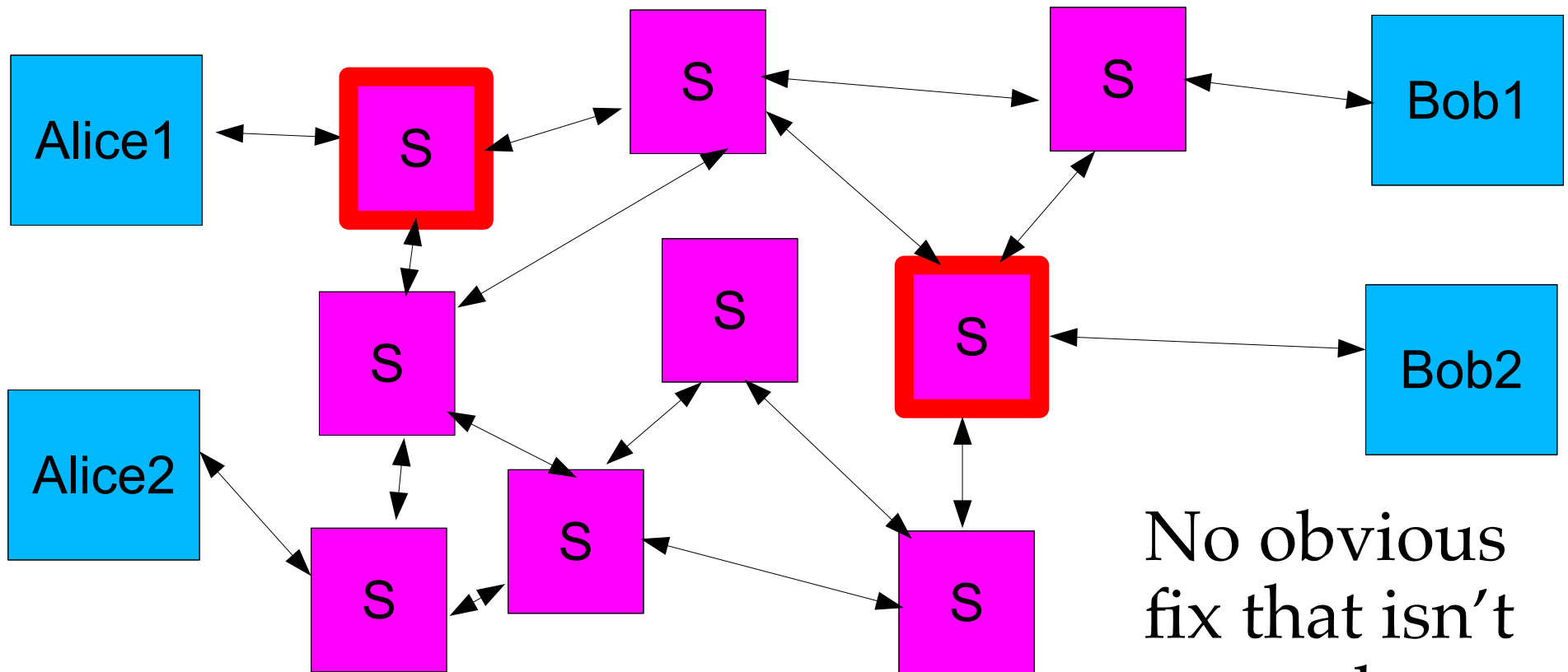
A hostile first hop can tell Alice is talking, but not to whom.



A hostile last hop can tell somebody is talking to Bob, but not who.



But: two hostile hops can correlate traffic patterns and link Alice to Bob.



No obvious fix that isn't extra-slow.

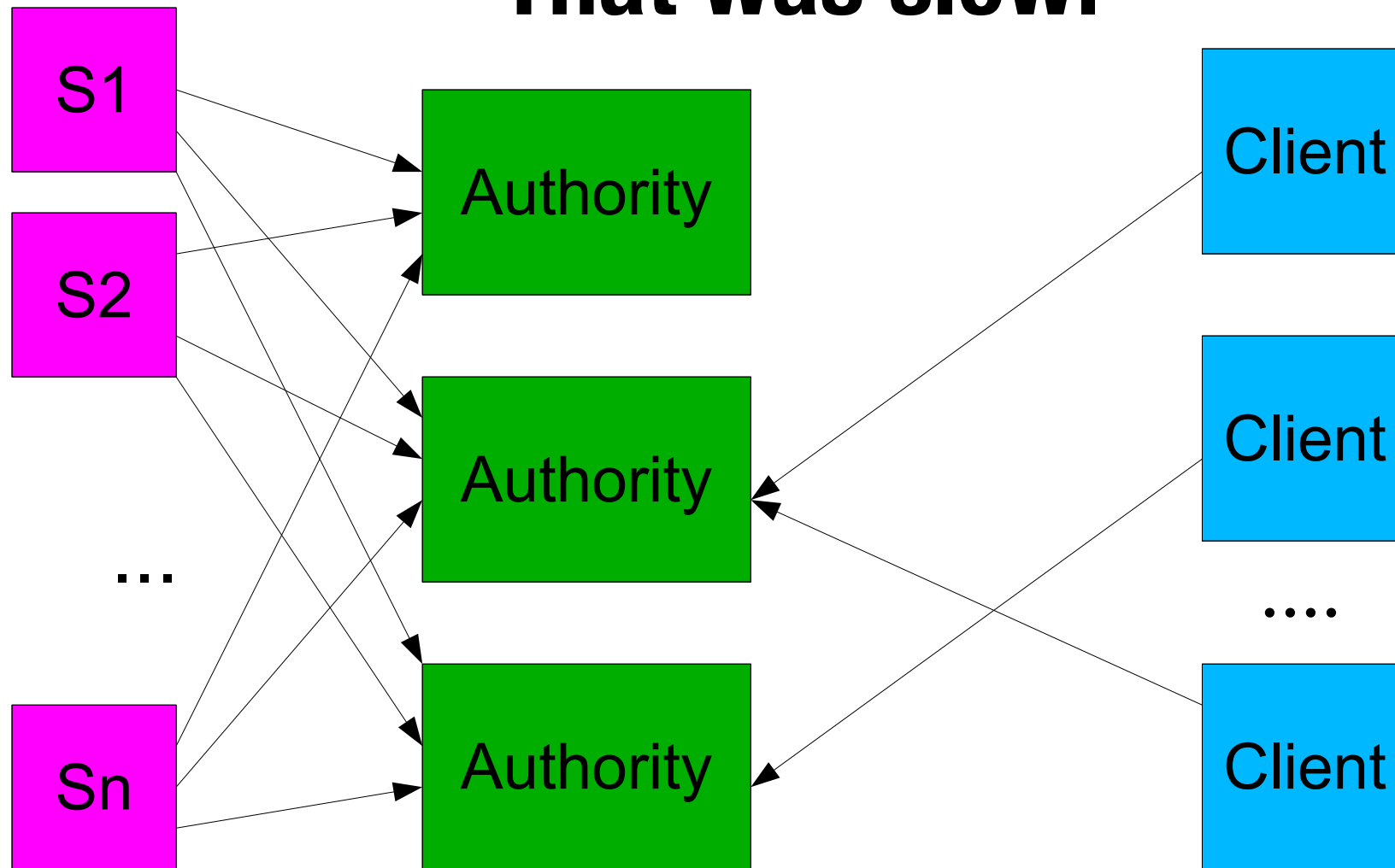
I. Directories and server discovery

We need to tell clients about servers.

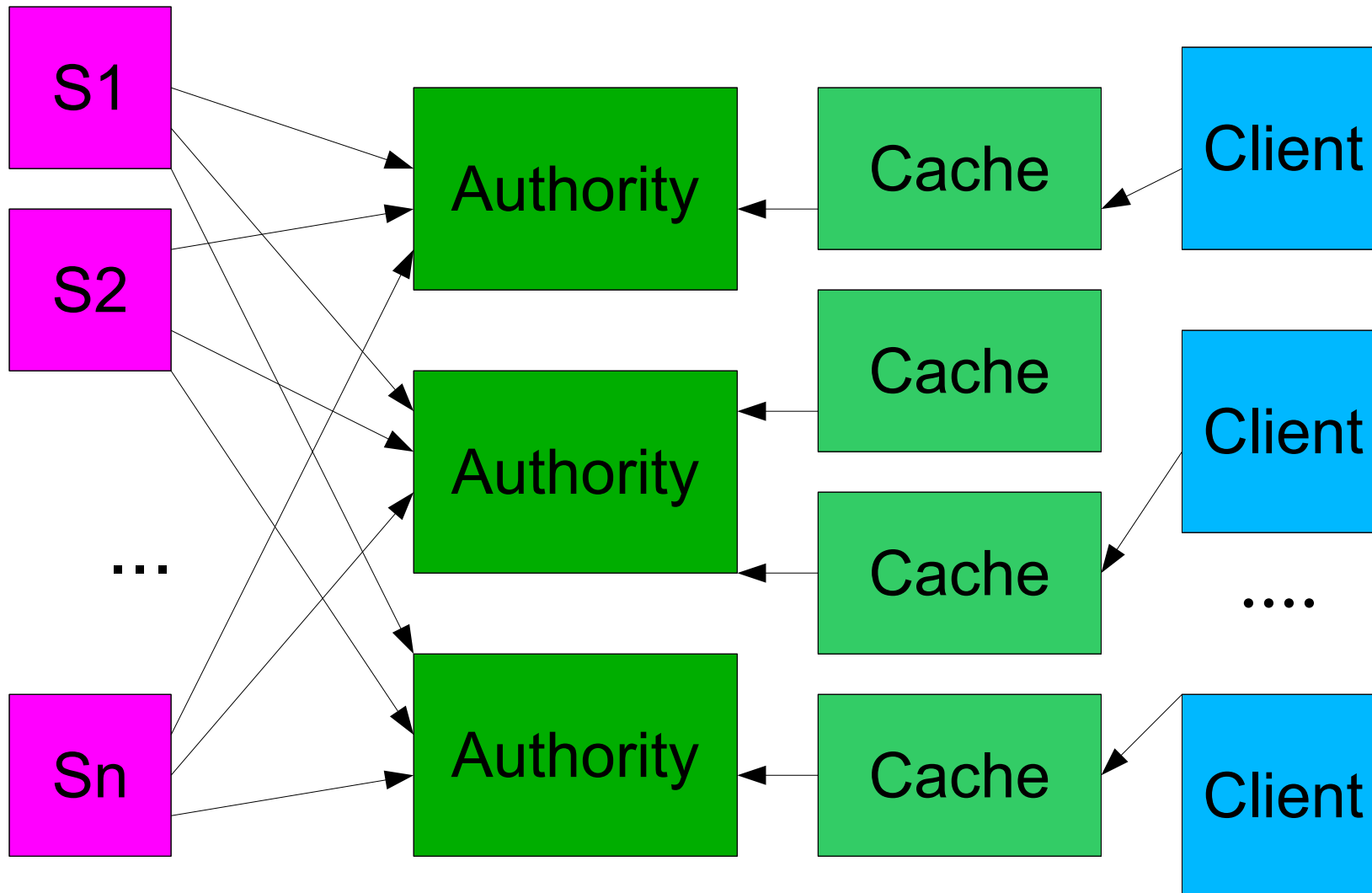
- Every client must know *every* server.
 - (If you just ask a server for a list of neighbors, it can trivially lie.)
- All clients must know the *same* servers.
- Servers shouldn't be able to impersonate each other.
 - (Use self-signed descriptions; identity by PK.)
- Bandwidth matters a lot.

2004: every authority published a big list of server information.

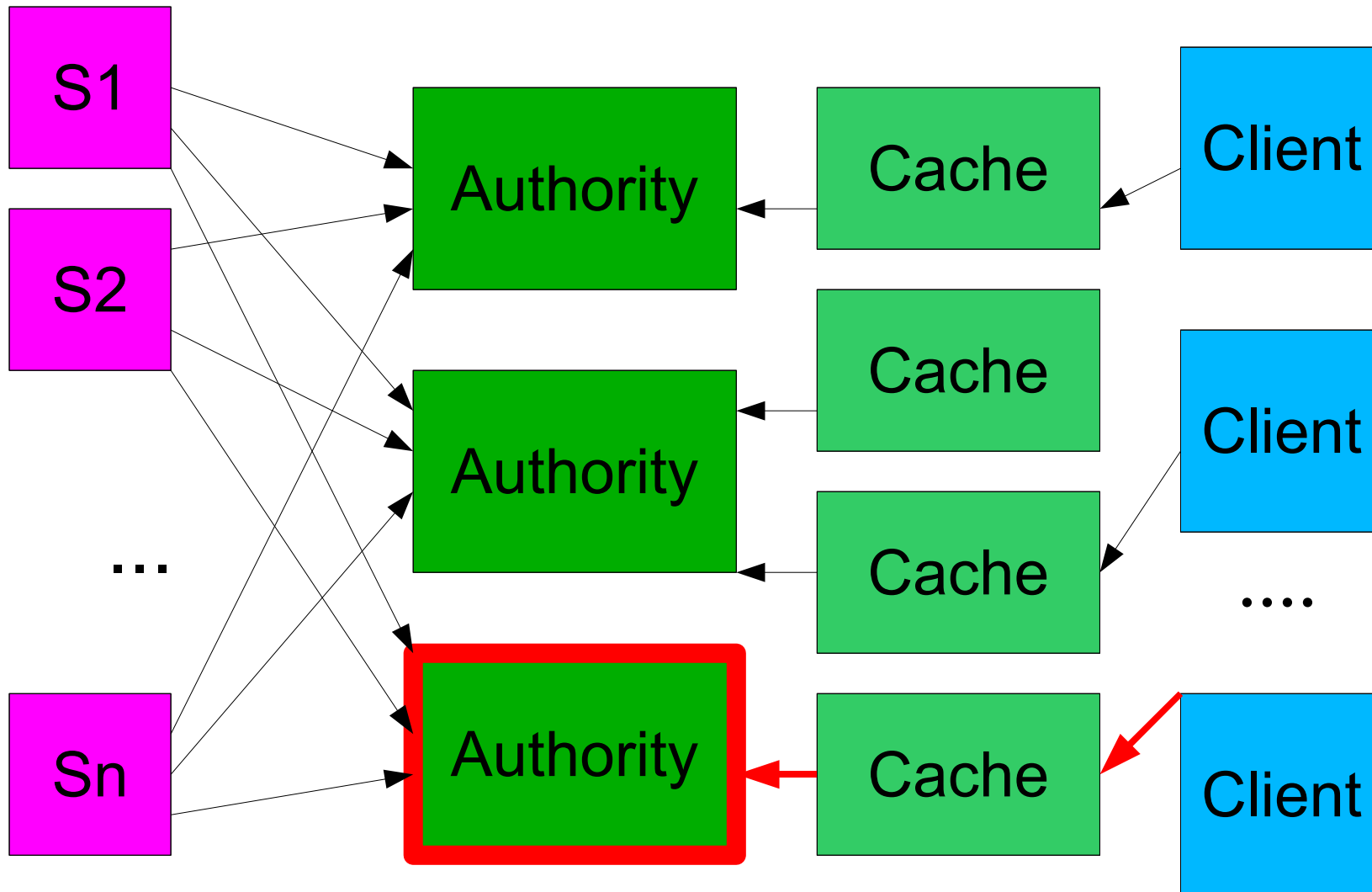
That was slow.



Adding caches helped with performance...



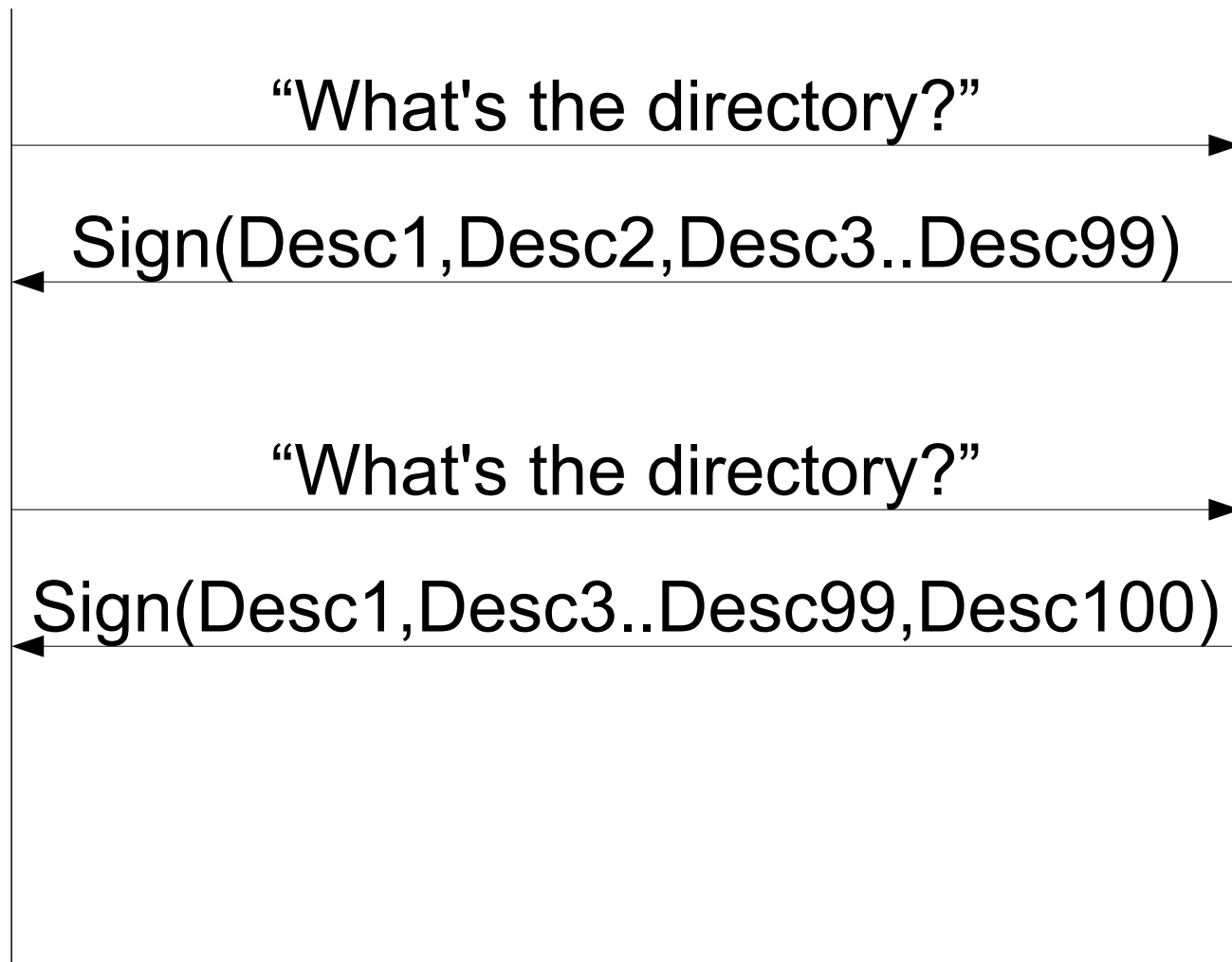
But a single bad authority could still break clients badly...



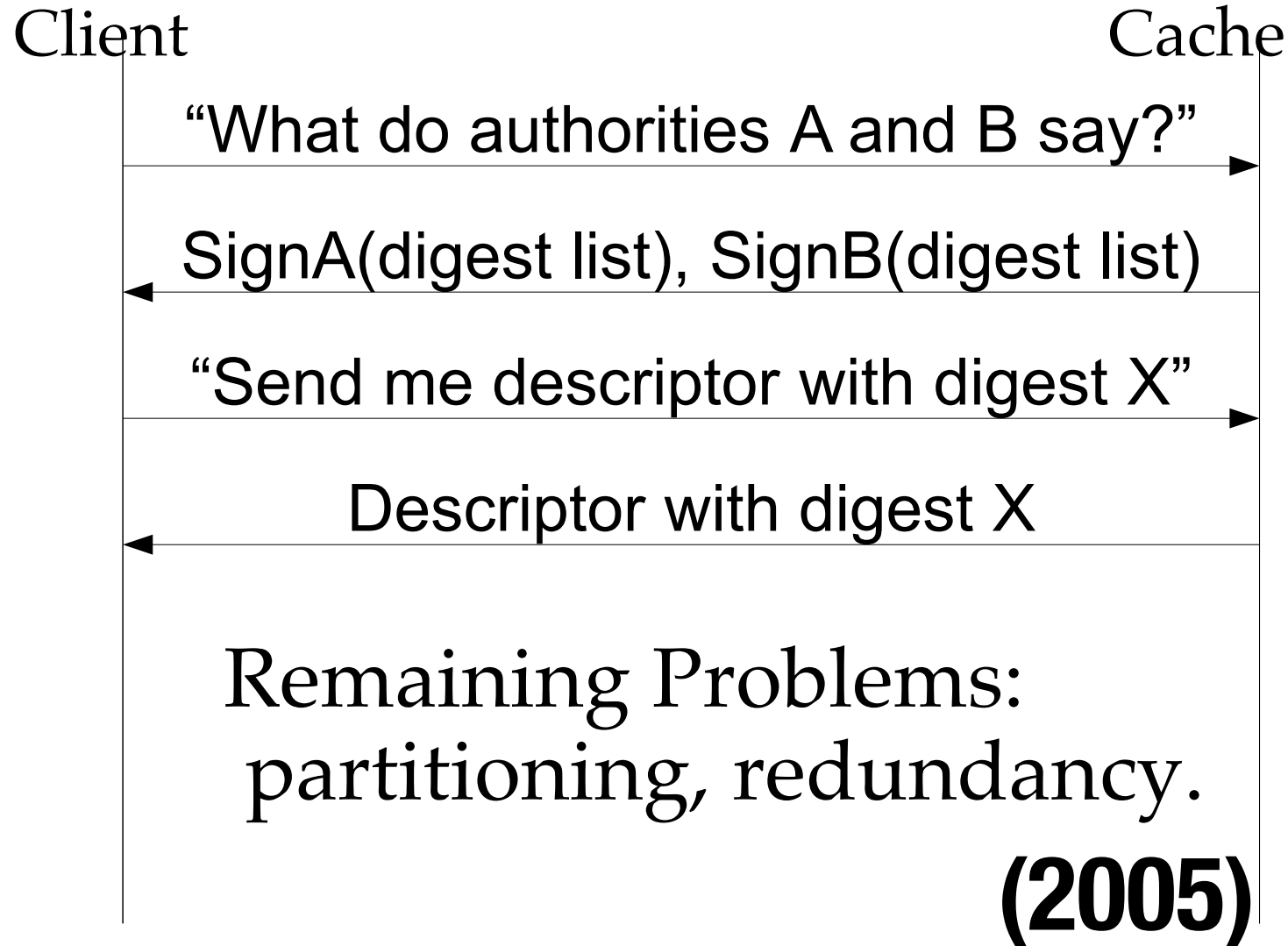
And most information was redundant.

Client

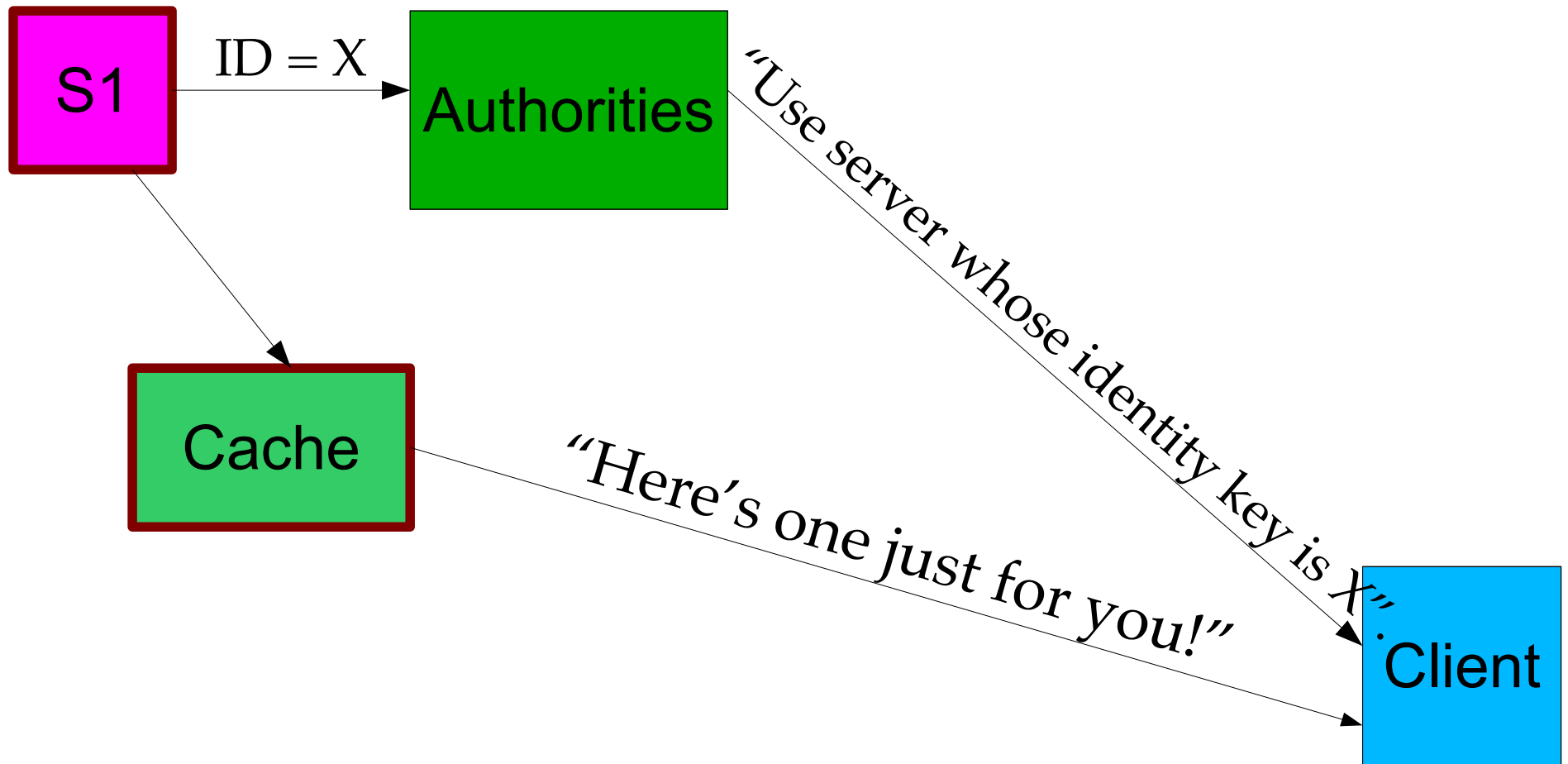
Cache



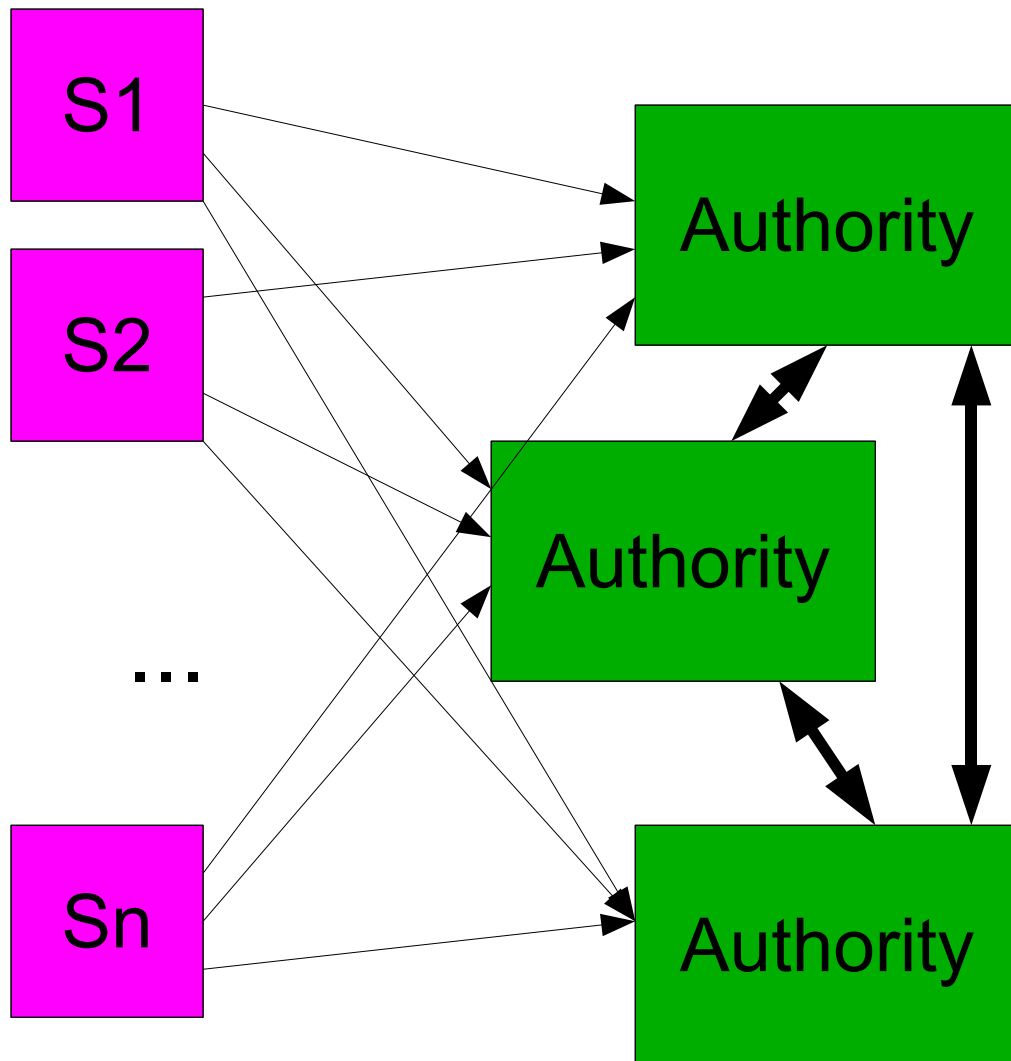
So split directory into status (signed) and individual descriptors



Naming and requesting descriptors by digest prevents attacks.



Authorities now vote on a single consensus status document.



1. Distribute signed opinions.
2. Compute result of vote, and sign it.
3. Distribute signatures; make multi-signed document.
4. Clients check signatures.
5. Profit!

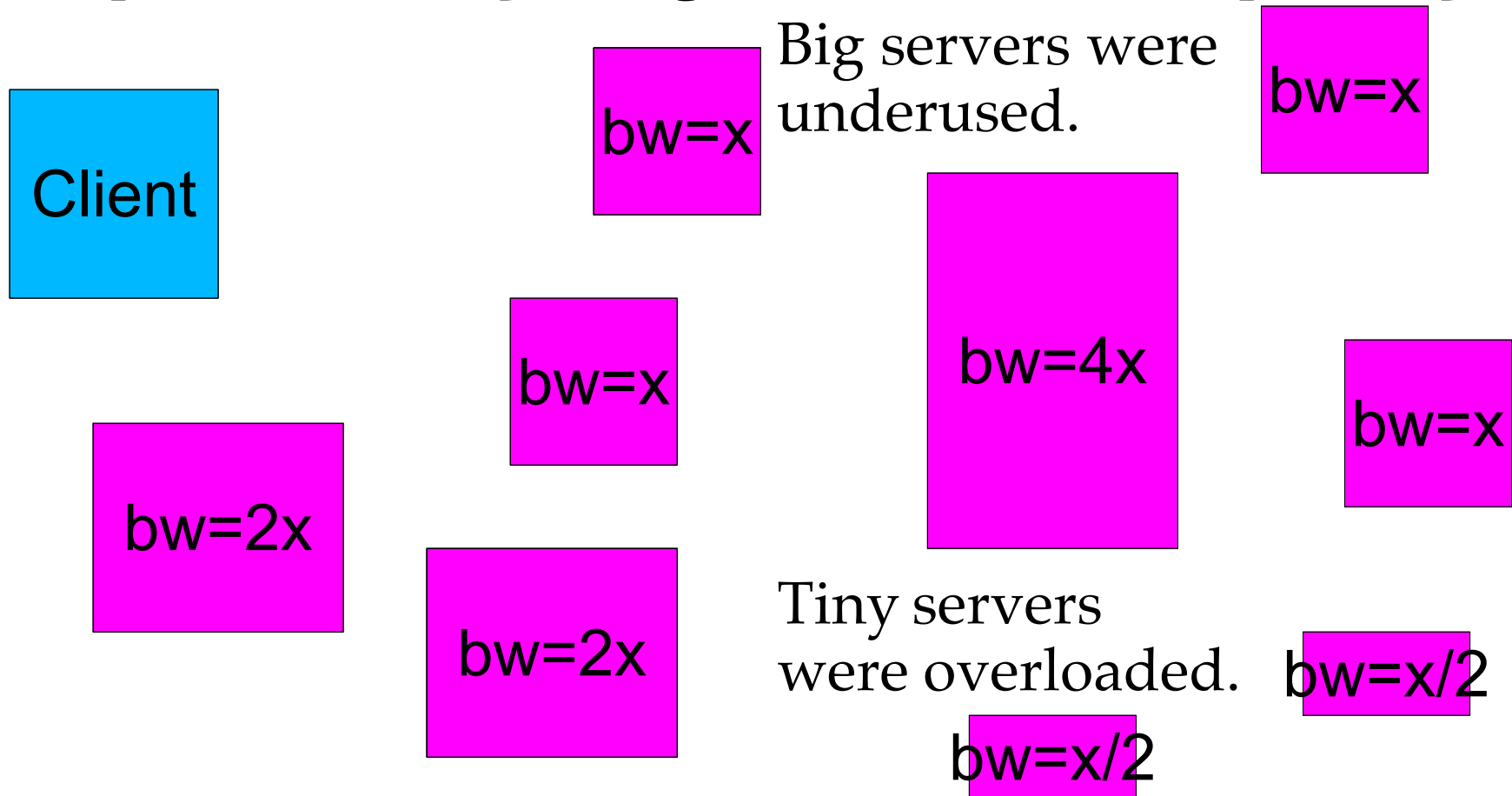
(2007)

Authorities say more than “yes/no” for each server.

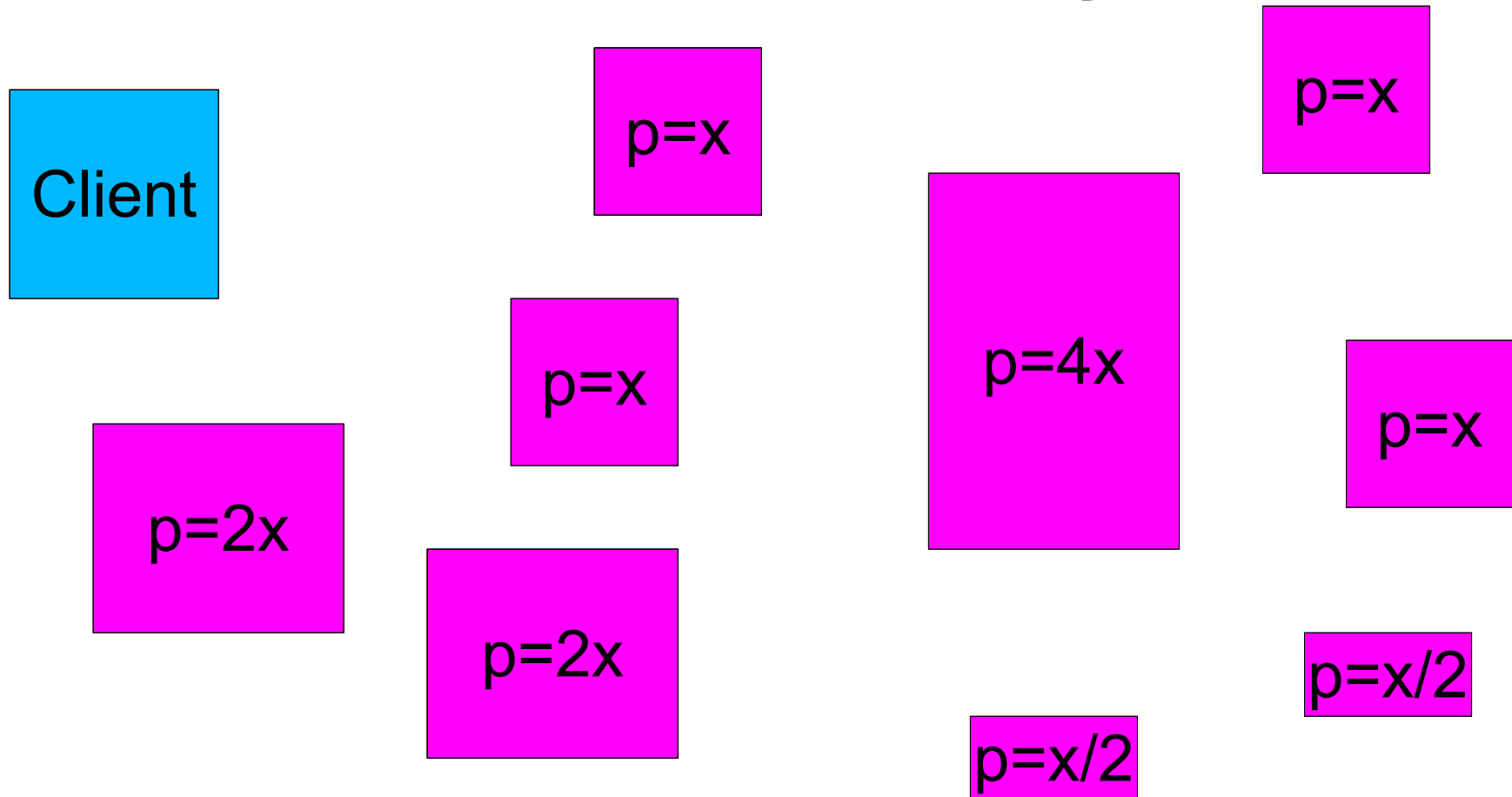
- Named? Authority?
- Running? Guard?
- Valid?
- Fast? (Actually determining these can be *hard*.)
- Stable?
- Bad exit? (Keywords define client behavior; authorities improve criteria.)
- Exit?

II. Path generation

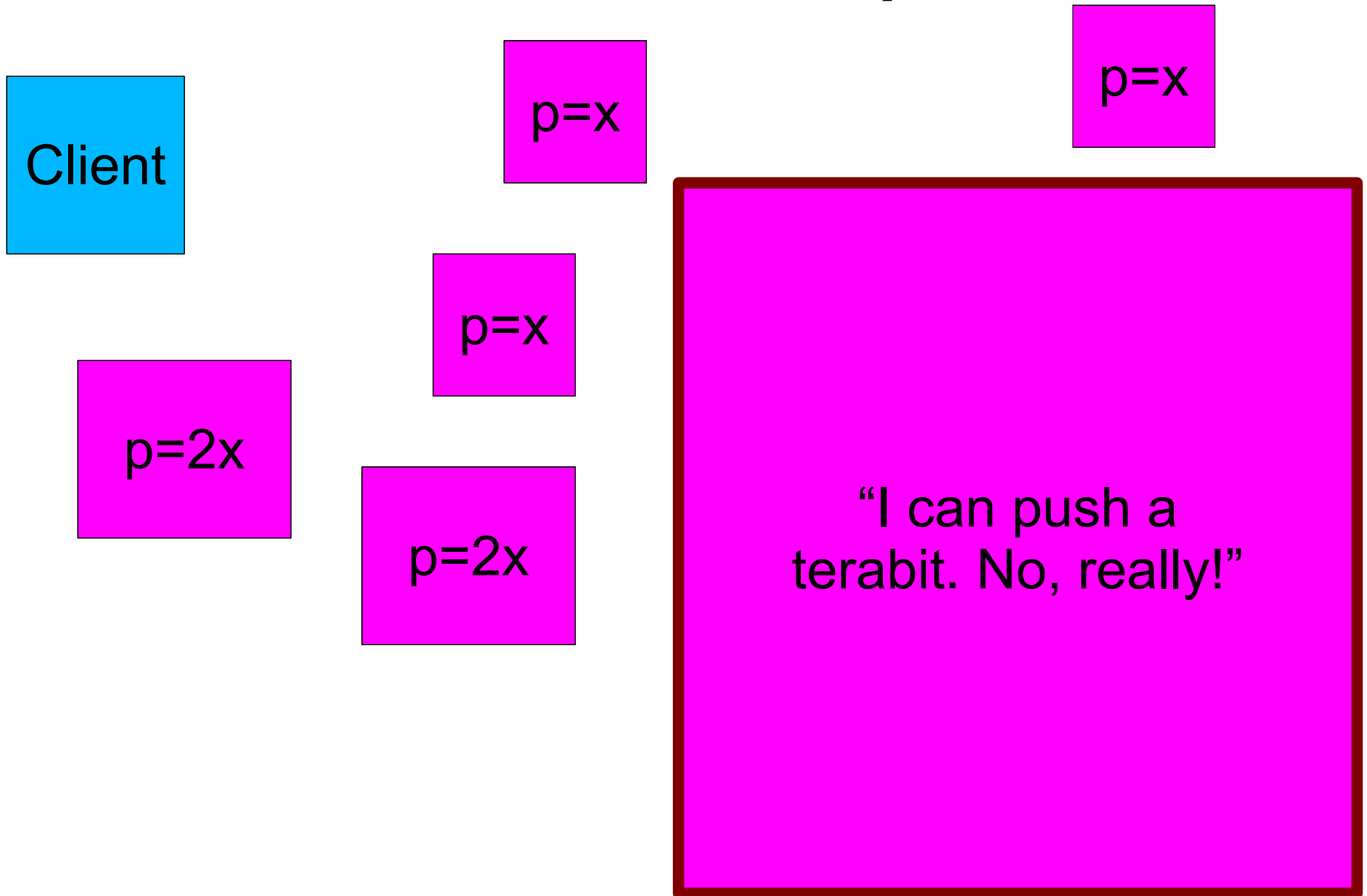
2004: all servers chosen with equal* probability, regardless of capacity.



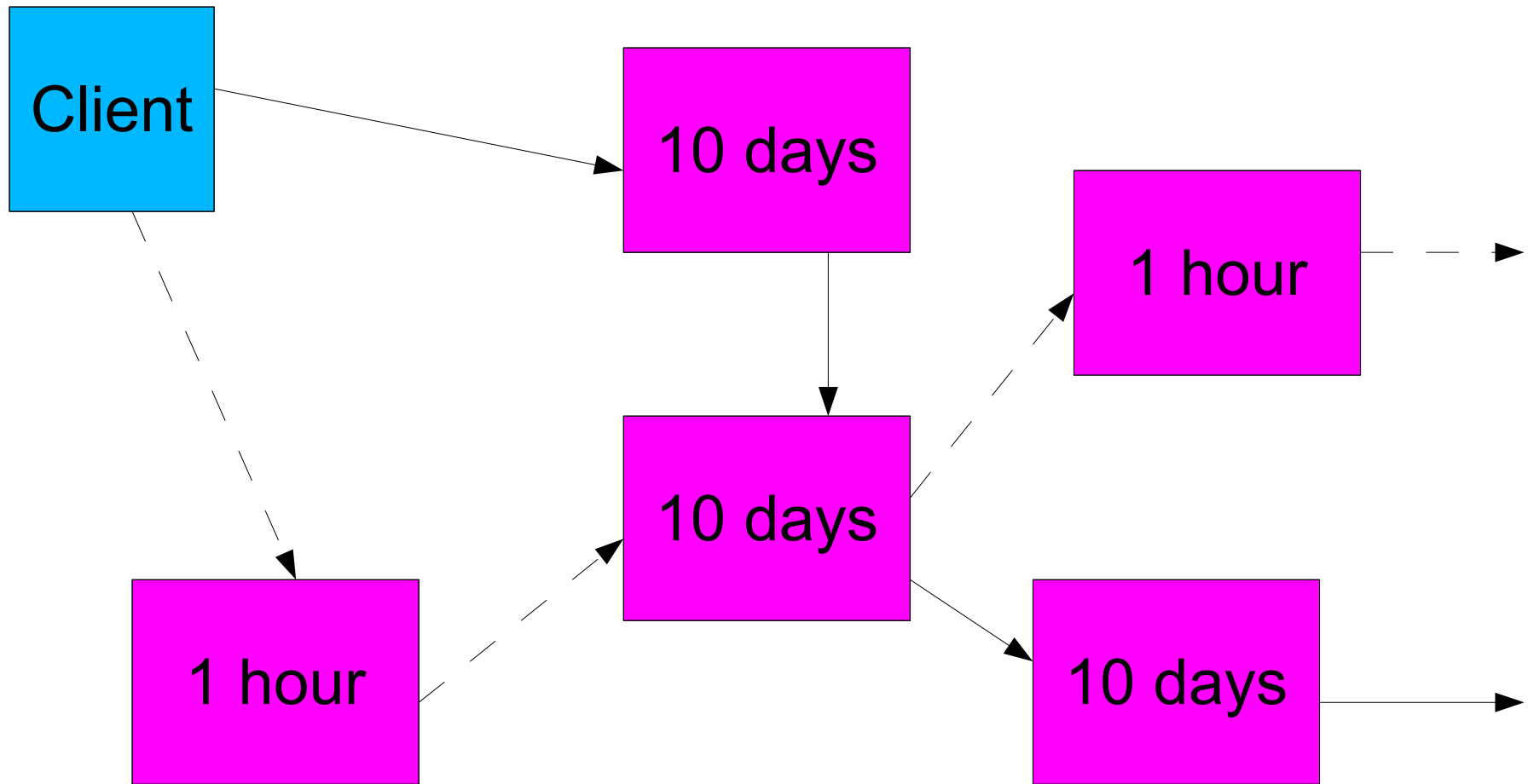
Now: Bandwidth is not uniform, so don't select uniformly.



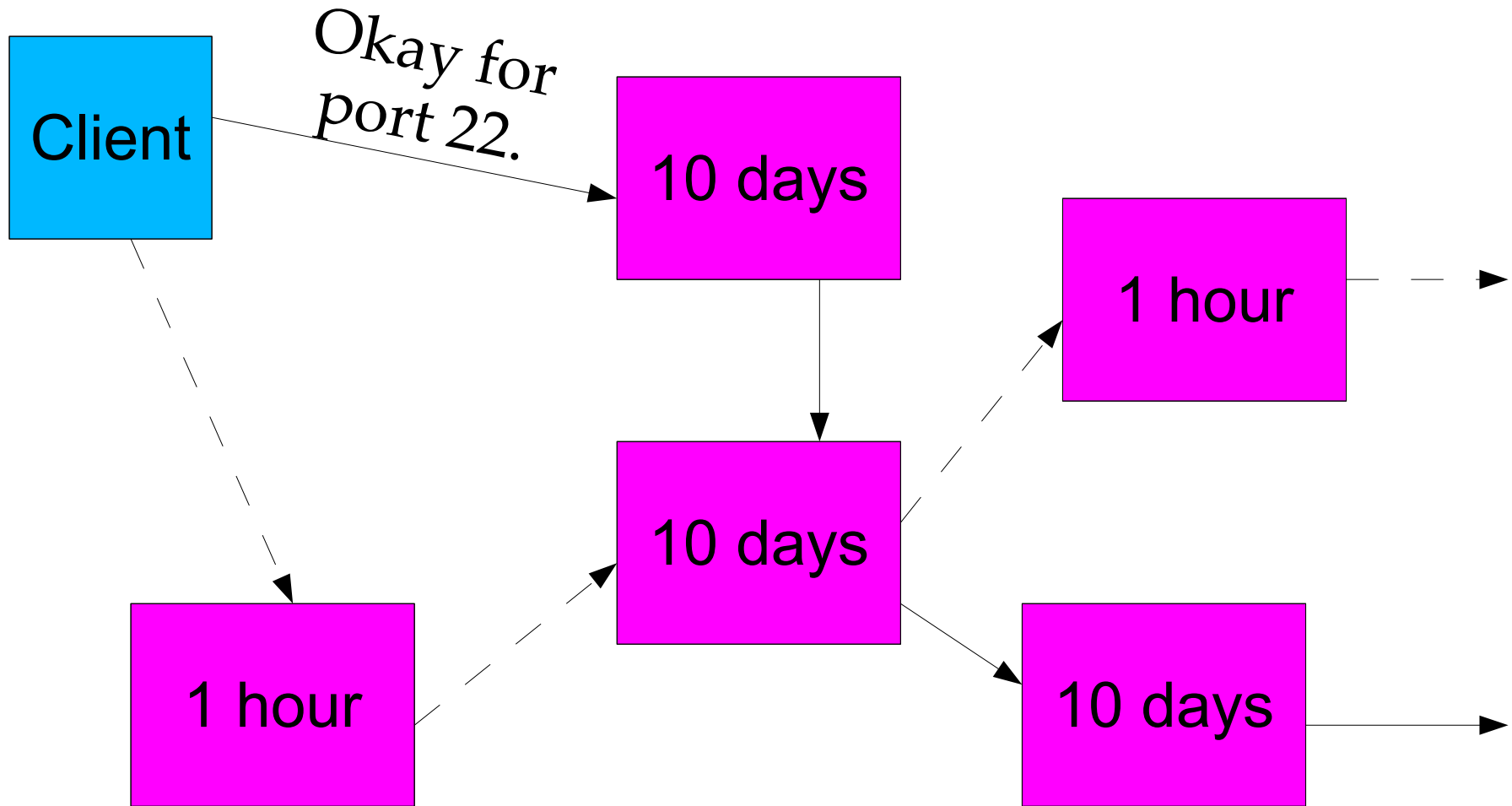
(But cap the maximum to prevent trust bottlenecks.)



Unstable servers are useful, but not for (SSH, IM, ...)



Use long-lived servers for long-lived connections.



Our original “random” path-selection approach made sure that every client would eventually be profiled.

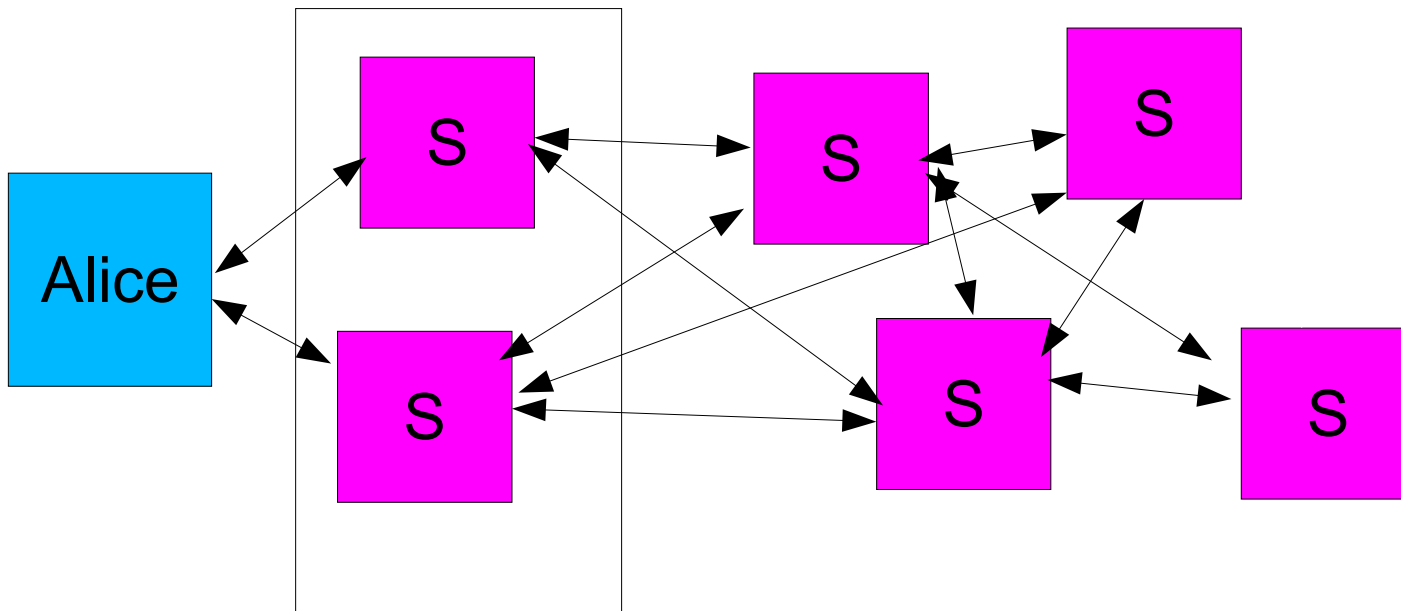
Alice loses if first and last hop are evil. (Correlation attacks)

Suppose c/n nodes (bandwidthwise) are compromised.

Therefore, $(c/n)^2$ of Alice's circuits are compromised.

Therefore, if Alice's behavior stays the same, she will eventually lose.

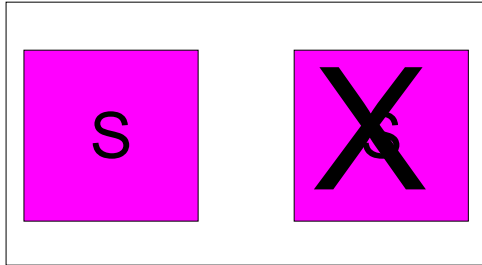
Tor clients now use “guard” servers to give long-term Alice a chance.



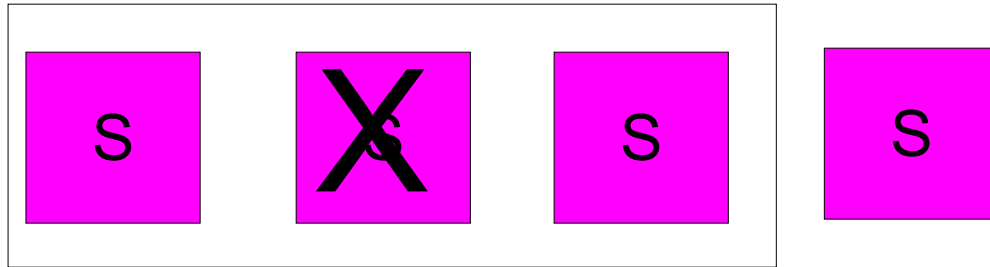
Chosen at random*, held fixed**.

If Alice's guards are good, Alice never has a vulnerable path.

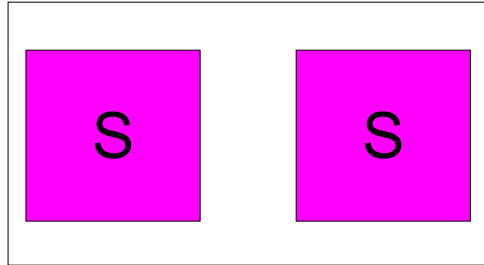
Okay, so guard nodes might go down.



**So add more as needed,
but keep them in order...**



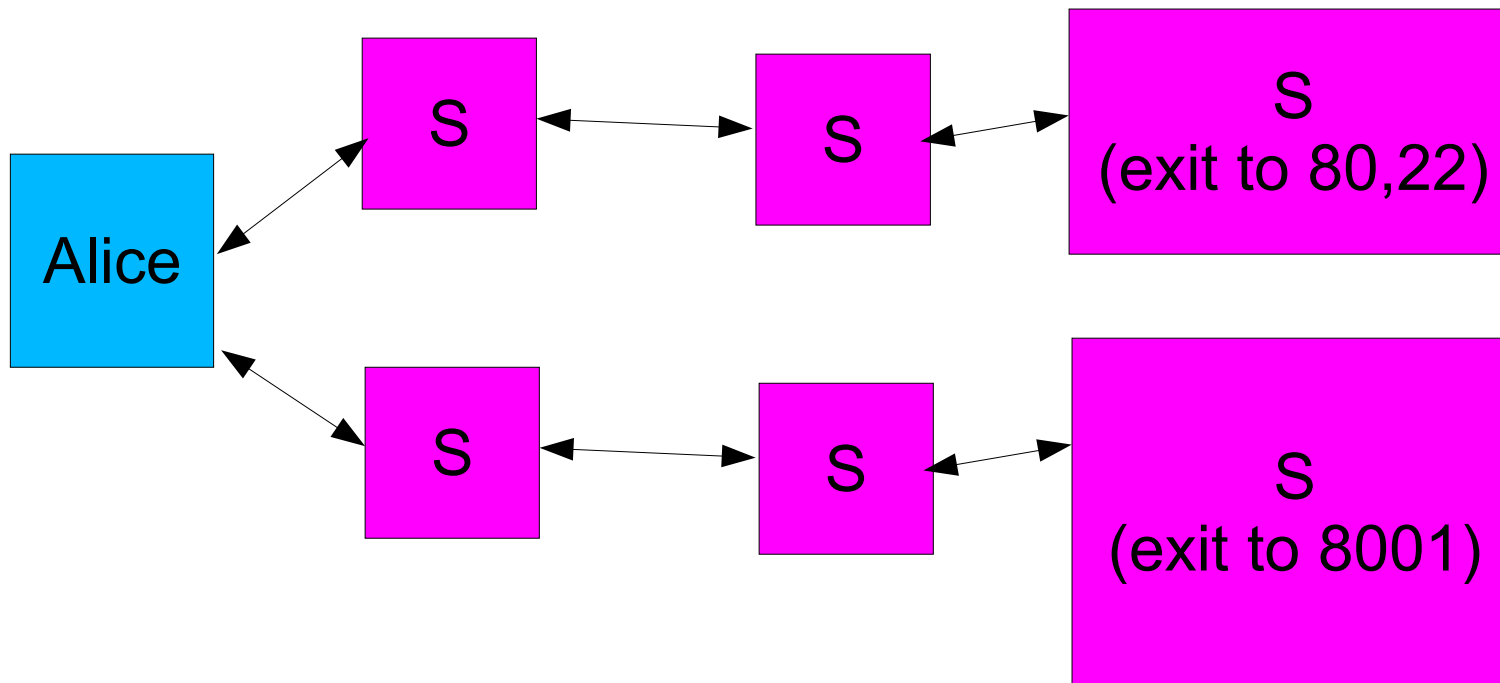
...so we can go back to the original set when they come back online.



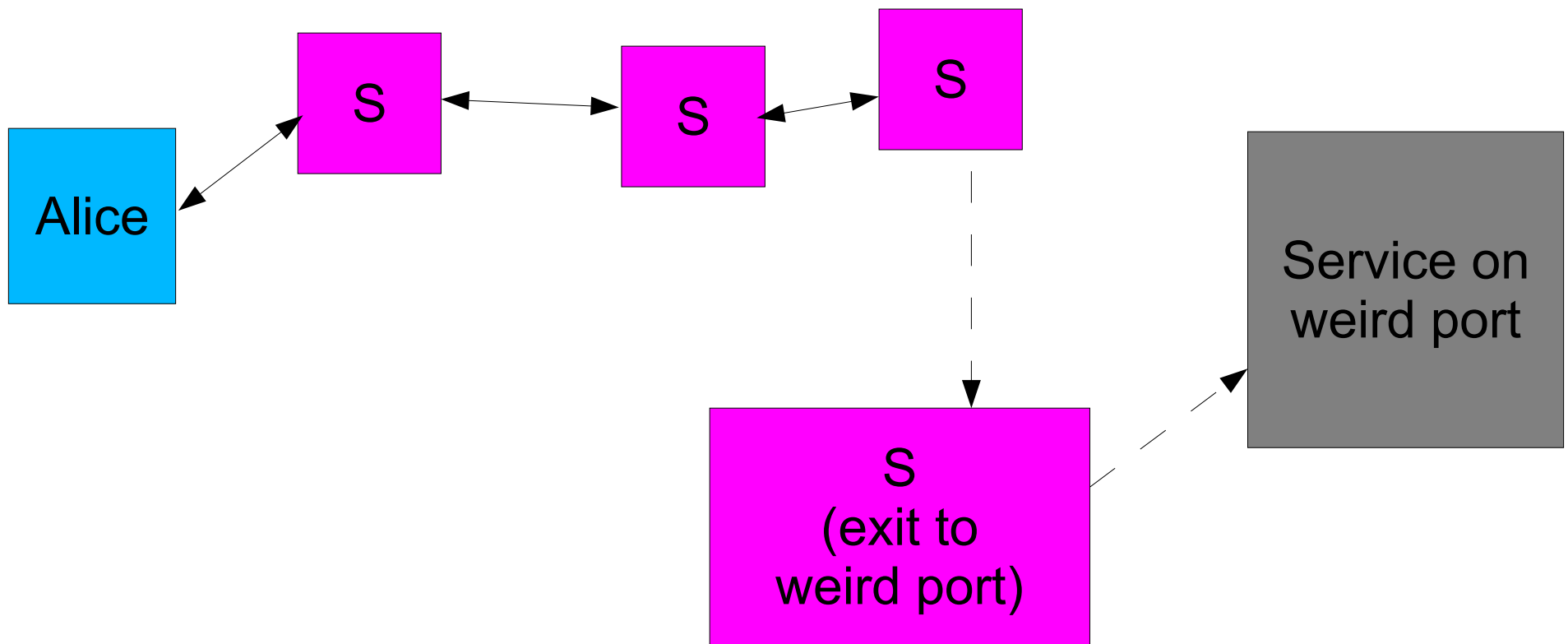
Old Tor: circuits built on-demand only.

This was slow.

Predict desired ports based on past behavior.

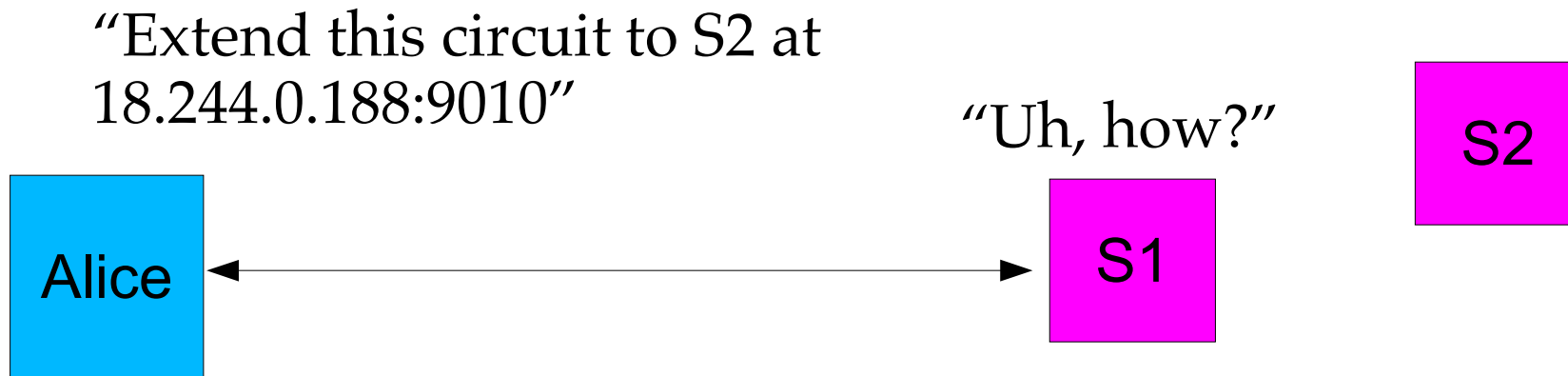


“Cannibalize” unused circuits for faster response to requests no circuit supports.



III. Circuit-building protocol

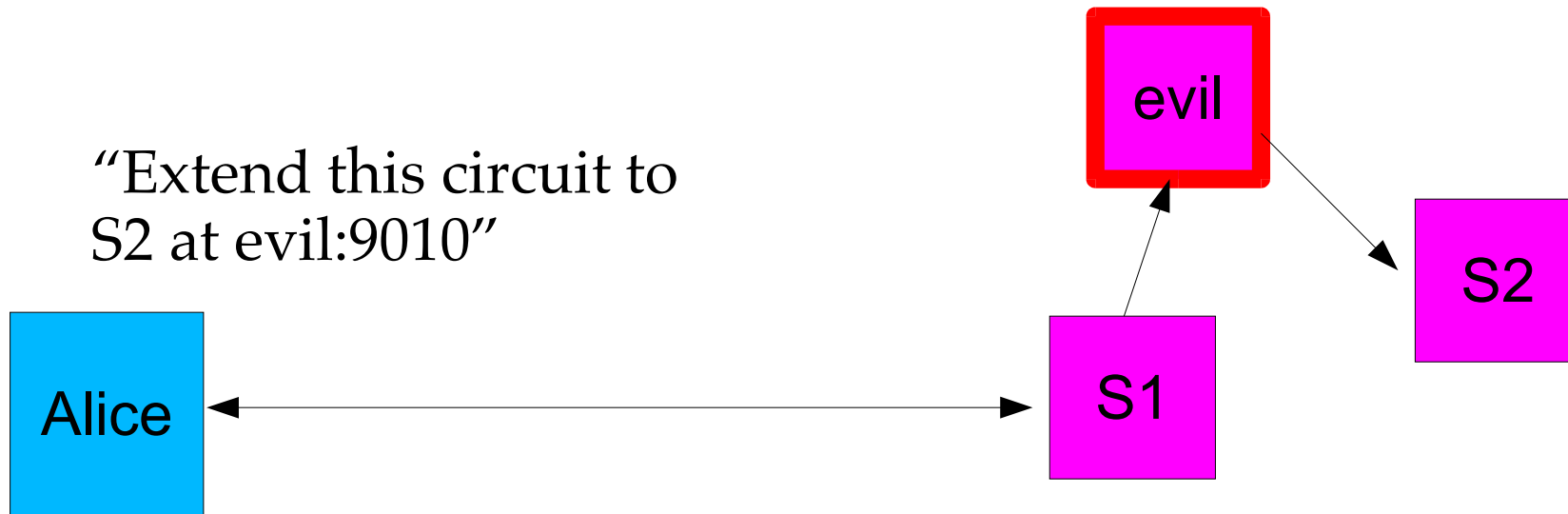
Extend by IP:Port was insufficient: nodes don't all know each other.



In practice, server knowledge is not 100% synchronized.

So, use identity key and IP.

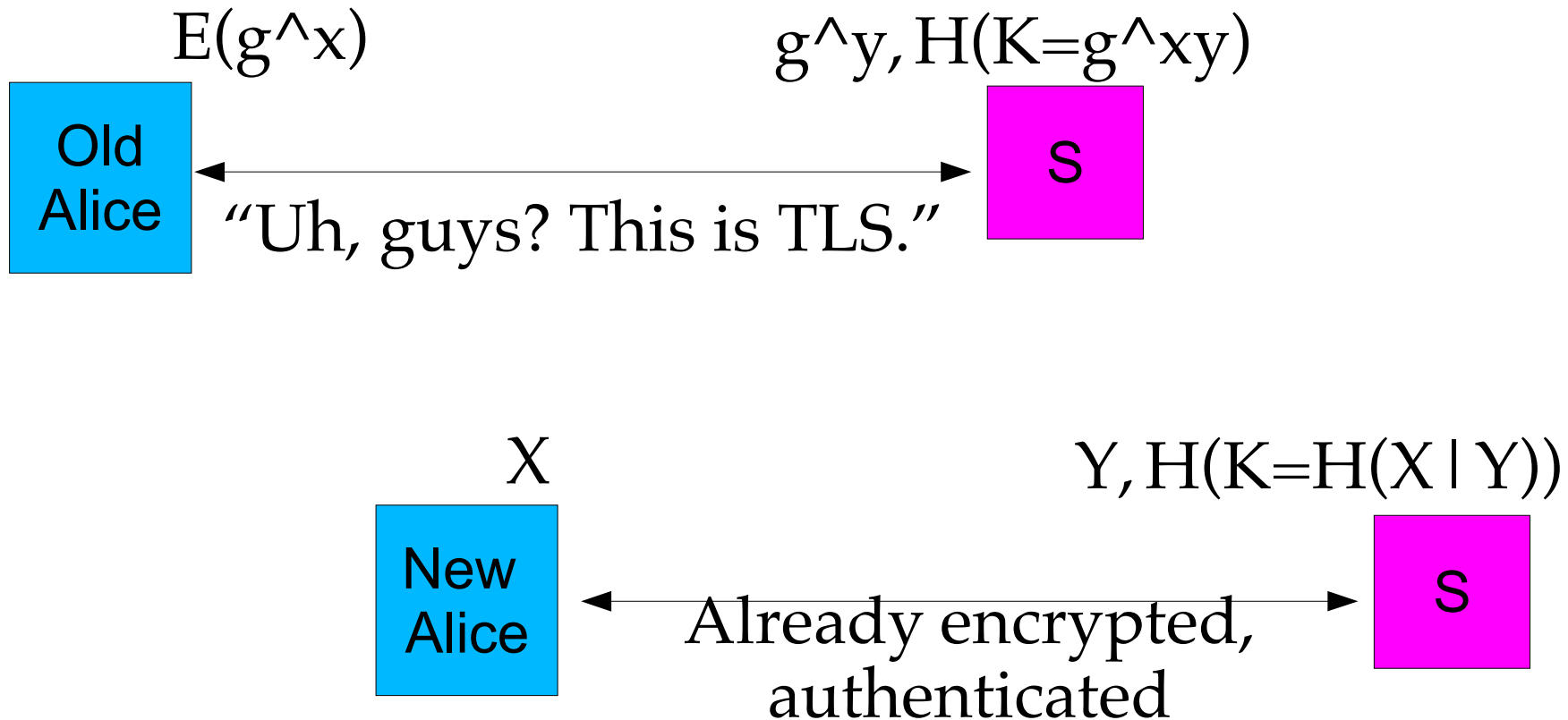
Using key-only ID for this created an MITM attack.



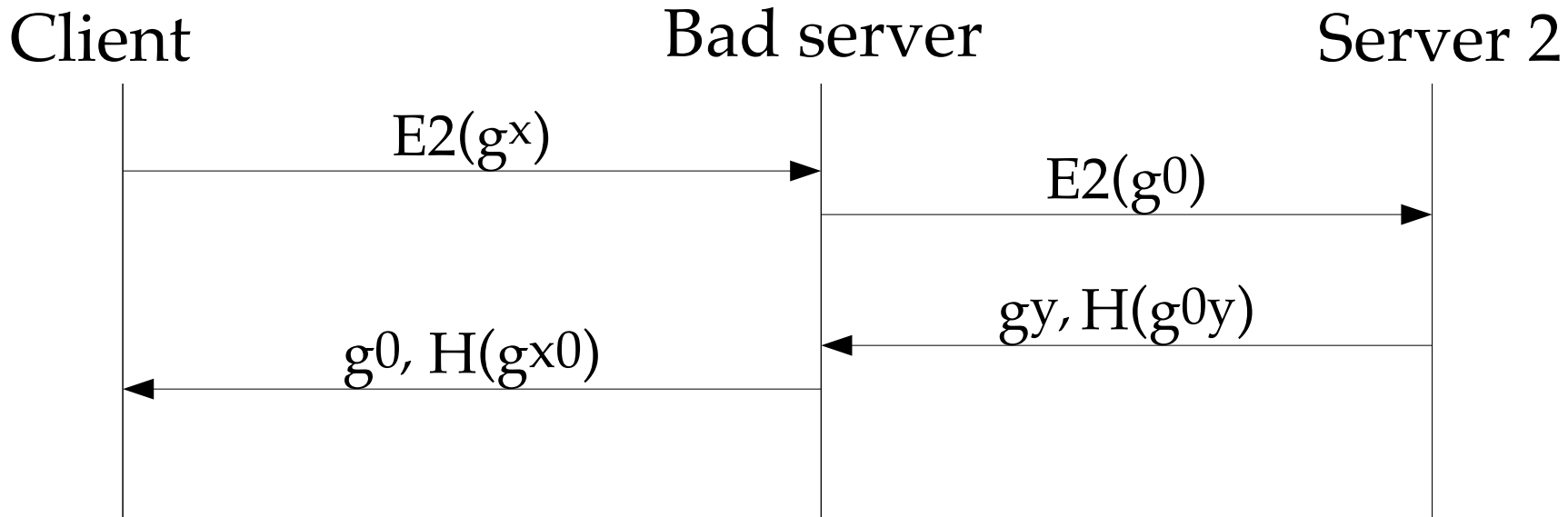
Only good for traffic analysis...
but other users were effective.

(So, don't use *only* identity key.)

Using encrypted create cell for first hop was needless crypto.



Speaking of cryptography, check for bad values of g^x, g^y .



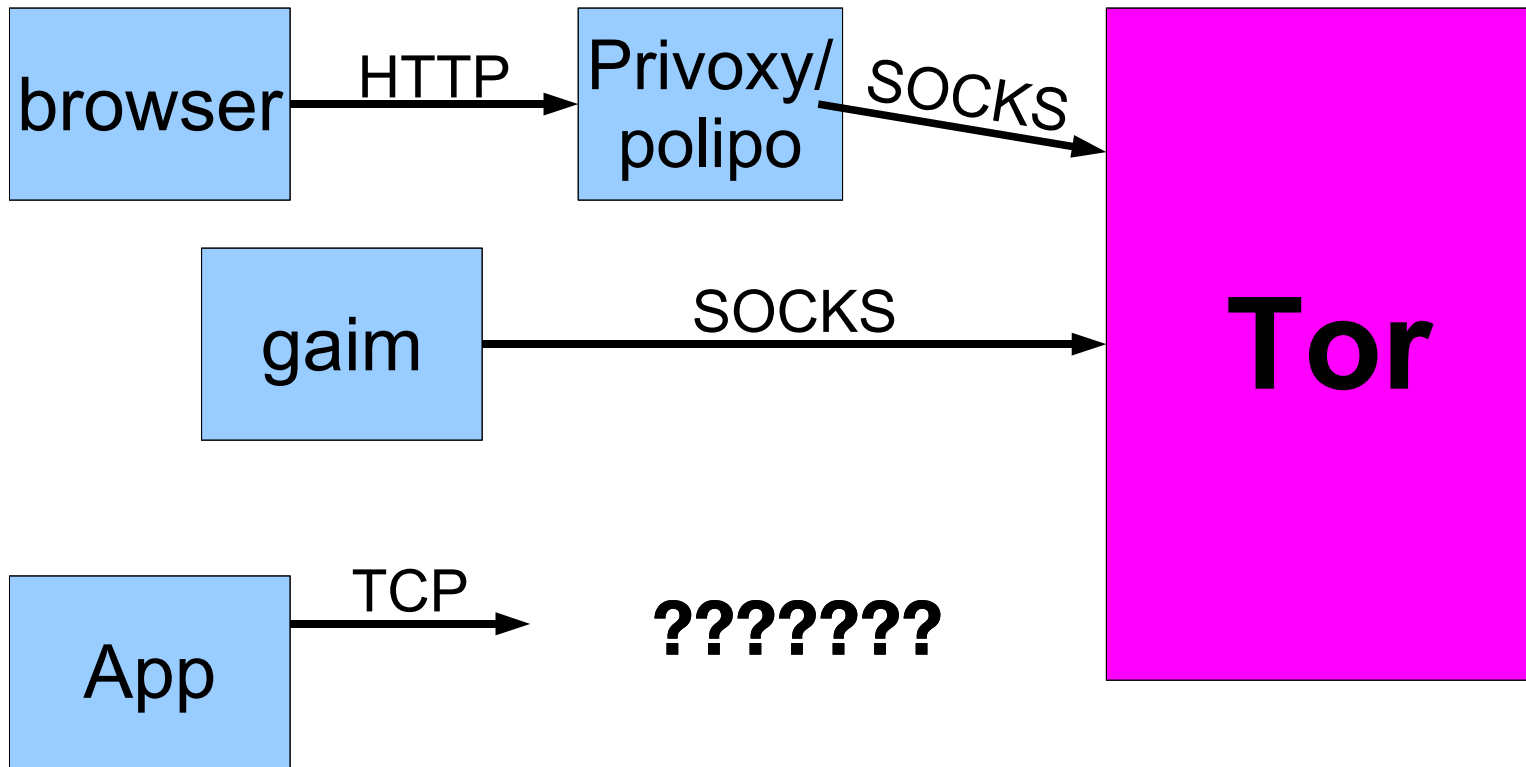
“oops.”

(Also, we patched OpenSSL for this.)

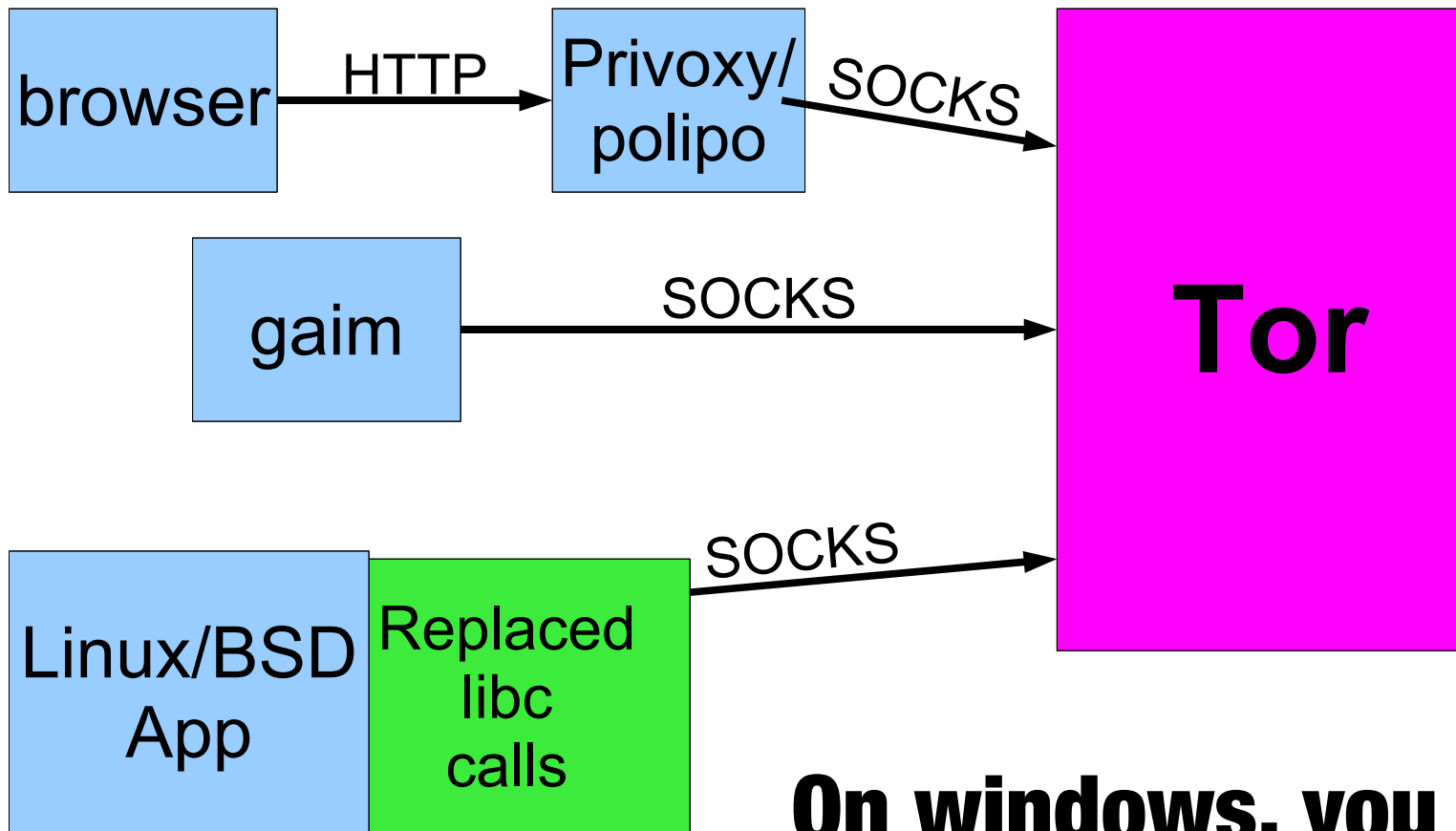
(but once we checked for bad g^x, g^y , Ian Goldberg could prove this protocol secure.)

III. Tools and features

Old Tor: everybody must speak SOCKS.

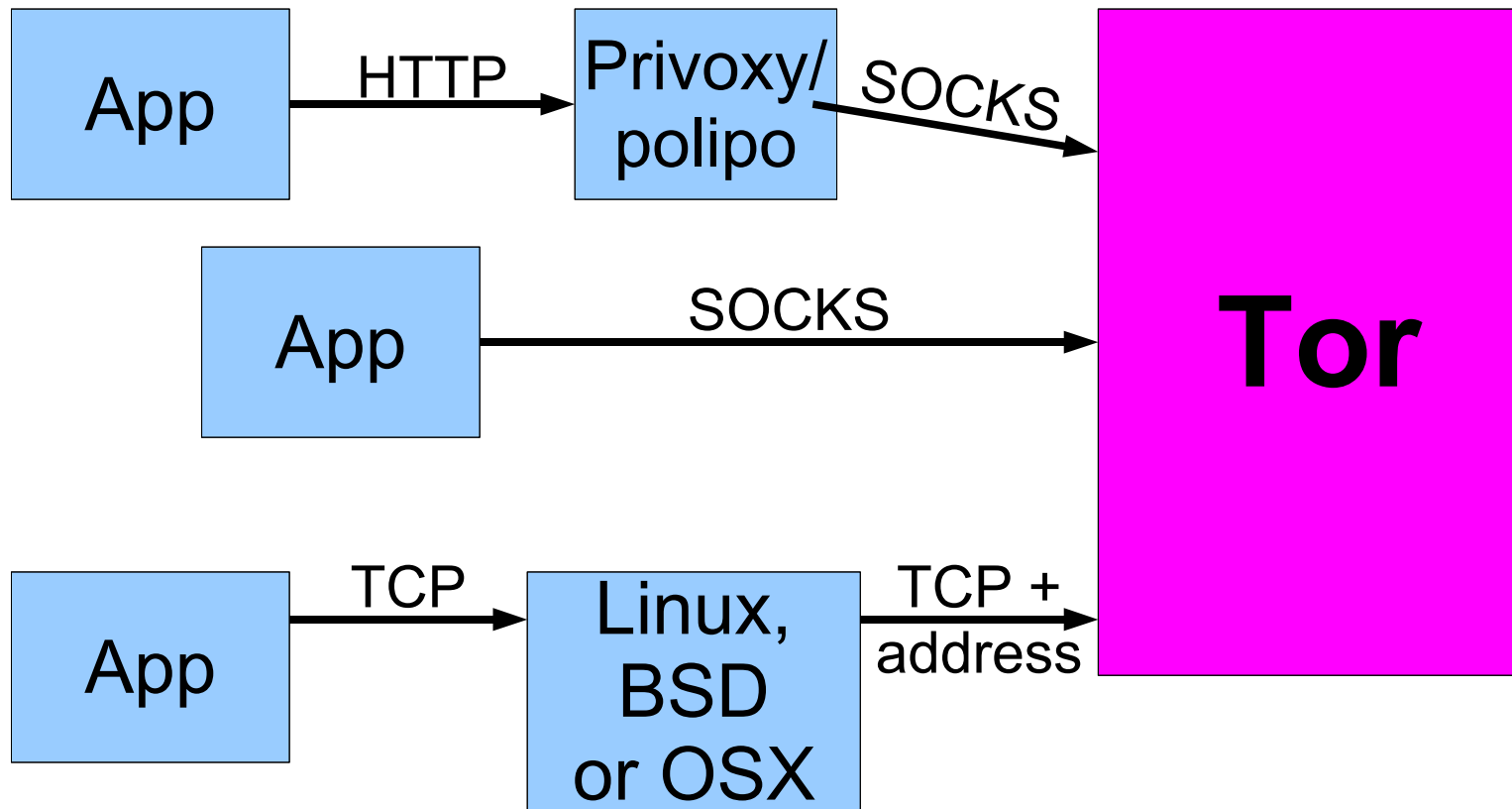


The old solutions kind of sucked.



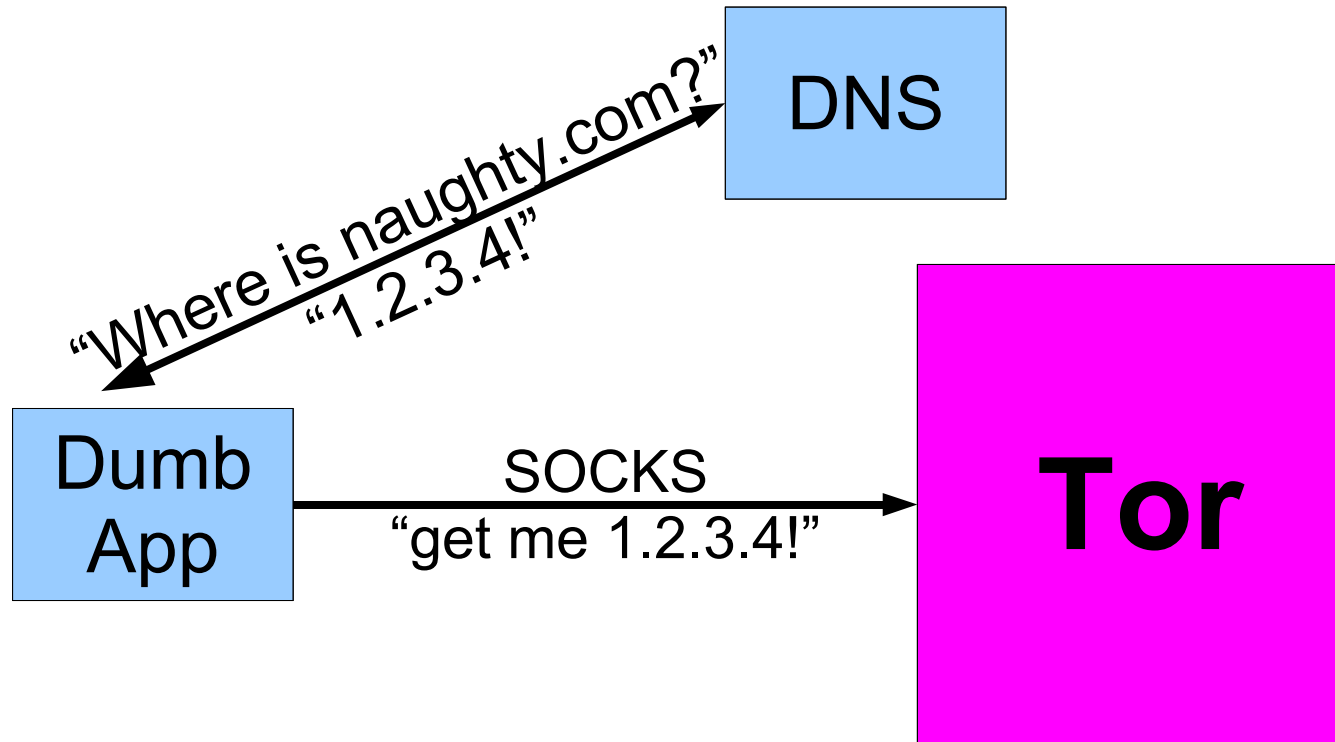
**On windows, you could
do a net driver...
OSX was screwed.**

TransPort (+iptables/pf) support any TCP



**You can also do use a VM as your router:
see JanusVM.**

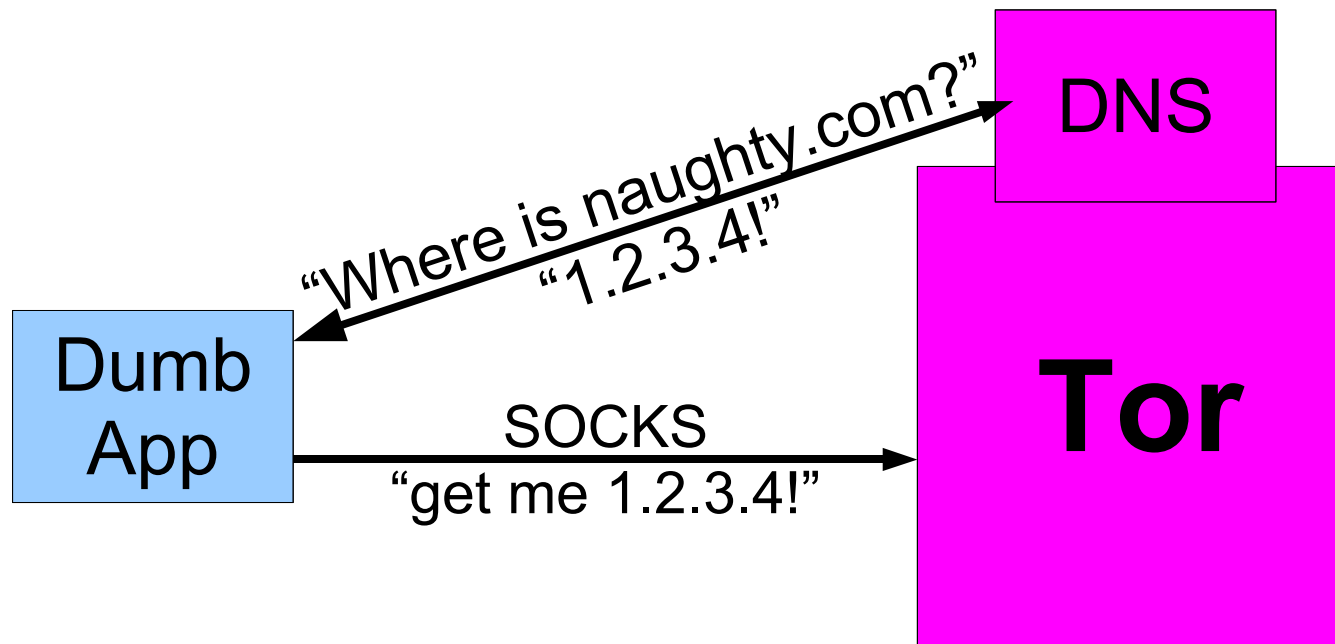
Problem: DNS leaks are hard to solve.



Old solution: “use SOCKS4a or else!”

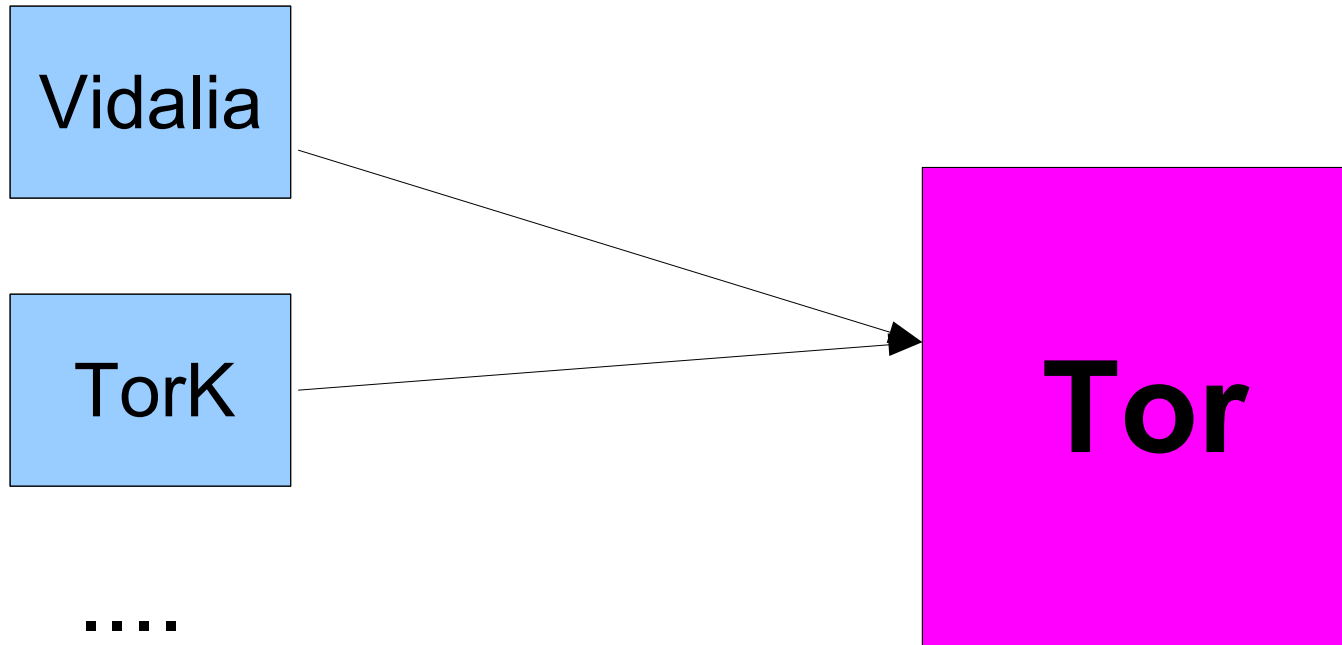


New solution: Tor acts as a DNS server



**This also lets dumb apps handle
.onion addresses.**

**Problem: editing text files is hard.
So, add support for external GUIs.**



Things to do:

- Tor: <https://torproject.org>
 - Try it out; want to run a server?
 - See docs and specs for more detail.
- Donate to Tor!
 - <https://torproject.org/donate.html>
 - (We're a tax-deductible charity!)
- Donate to EFF too!
 - I'm in the dunk tank at 6:30
- See more talks!
 - Roger at 2 on anti-censorship
 - Mike at 5 on securing the network and apps.