

# Tor performance problems ...and how to solve them

Roger Dingledine

The Tor Project

<https://www.torproject.org/>

# Tor: Big Picture

- Freely available (Open Source), unencumbered.
- Comes with a spec and full documentation:  
Dresden and Aachen implemented compatible Java Tor clients; researchers use it to study anonymity.
- 1500 active relays, 200000+ active users, >1Gbit/s.
- Official US 501(c)(3) nonprofit. Eight+ funded developers, dozens more dedicated volunteers.
- Funding from US DoD, Electronic Frontier Foundation, Voice of America, Human Rights Watch, Google, NLnet, ...you?

# **Anonymity serves different interests for different user groups.**

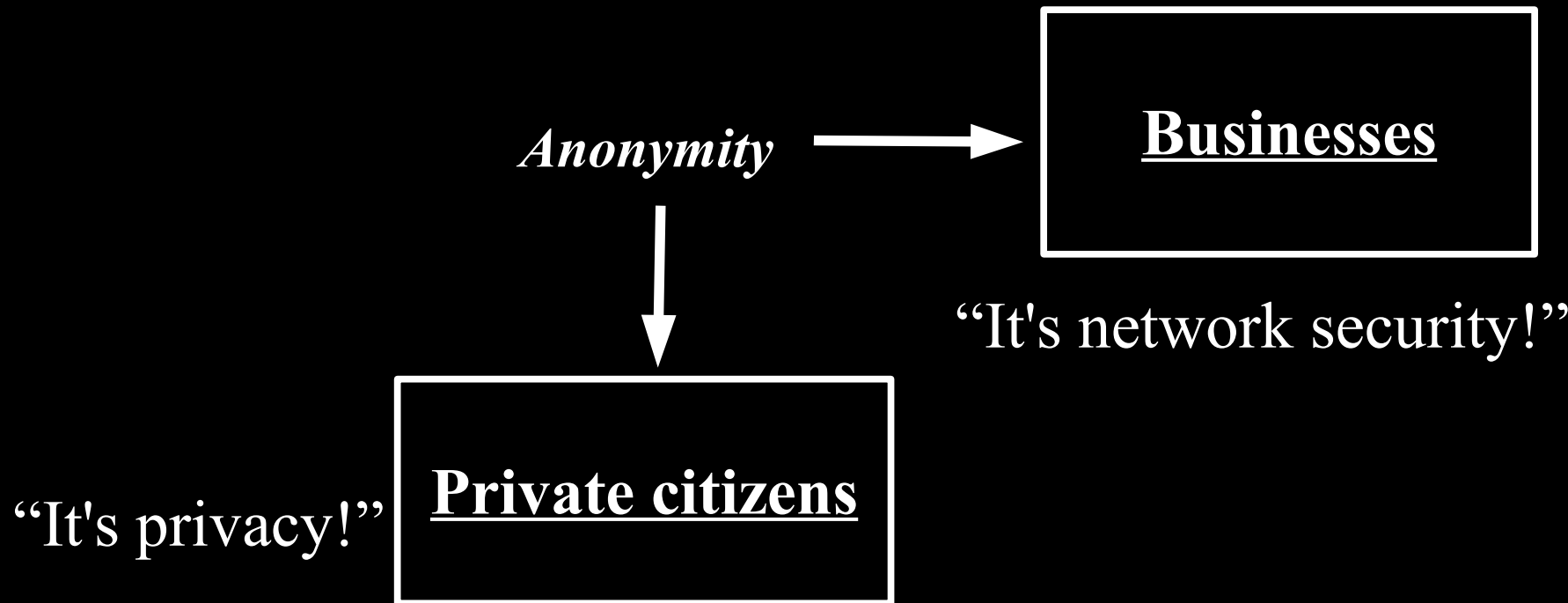
*Anonymity*



“It's privacy!”

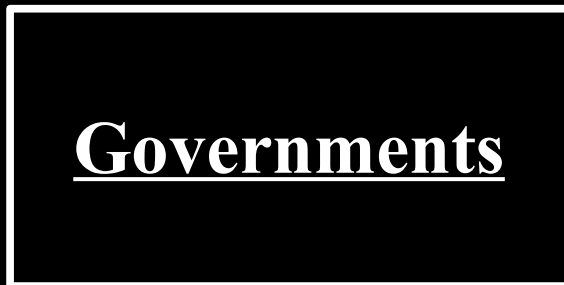
**Private citizens**

# Anonymity serves different interests for different user groups.



# Anonymity serves different interests for different user groups.

“It's traffic-analysis resistance!”



*Anonymity*

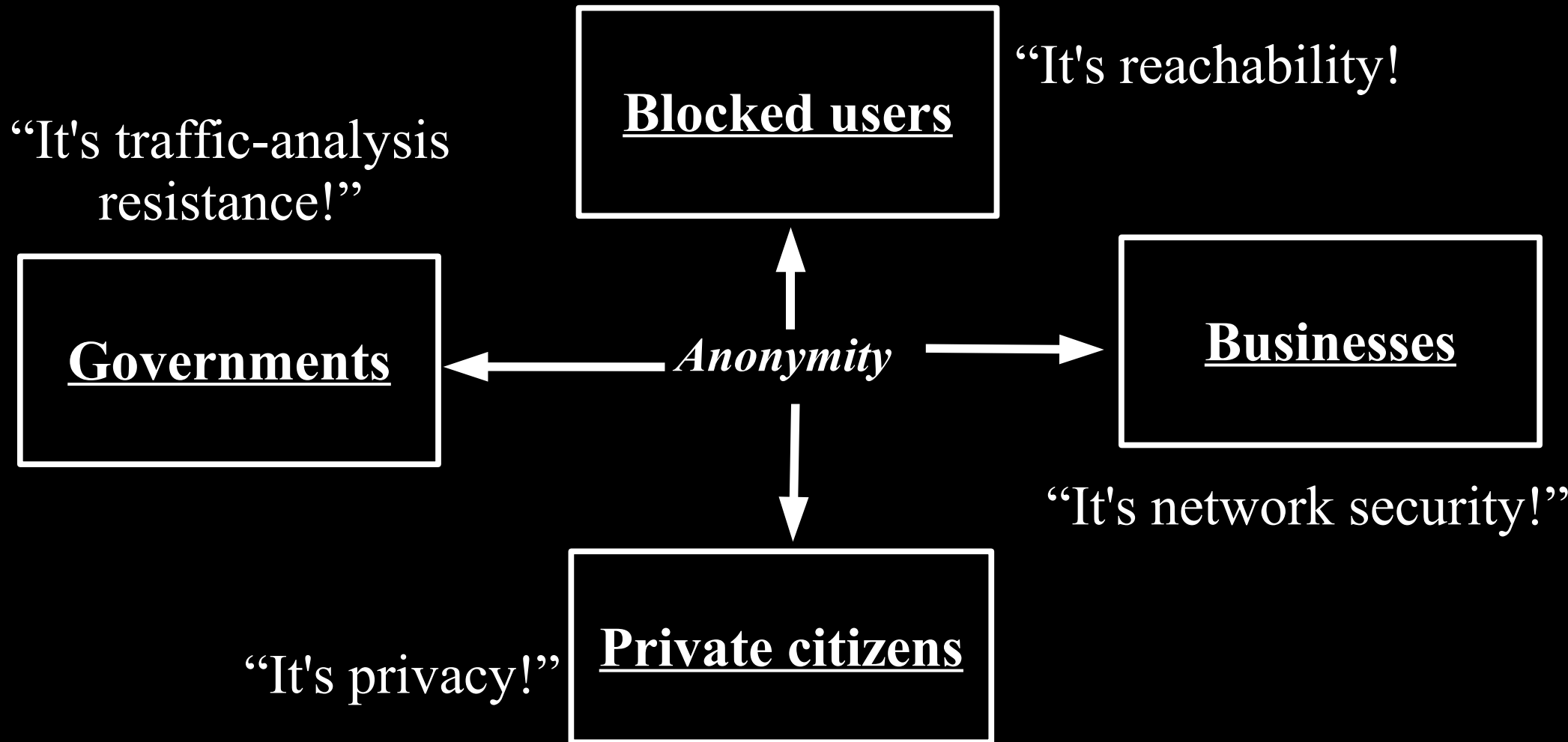


“It's network security!”

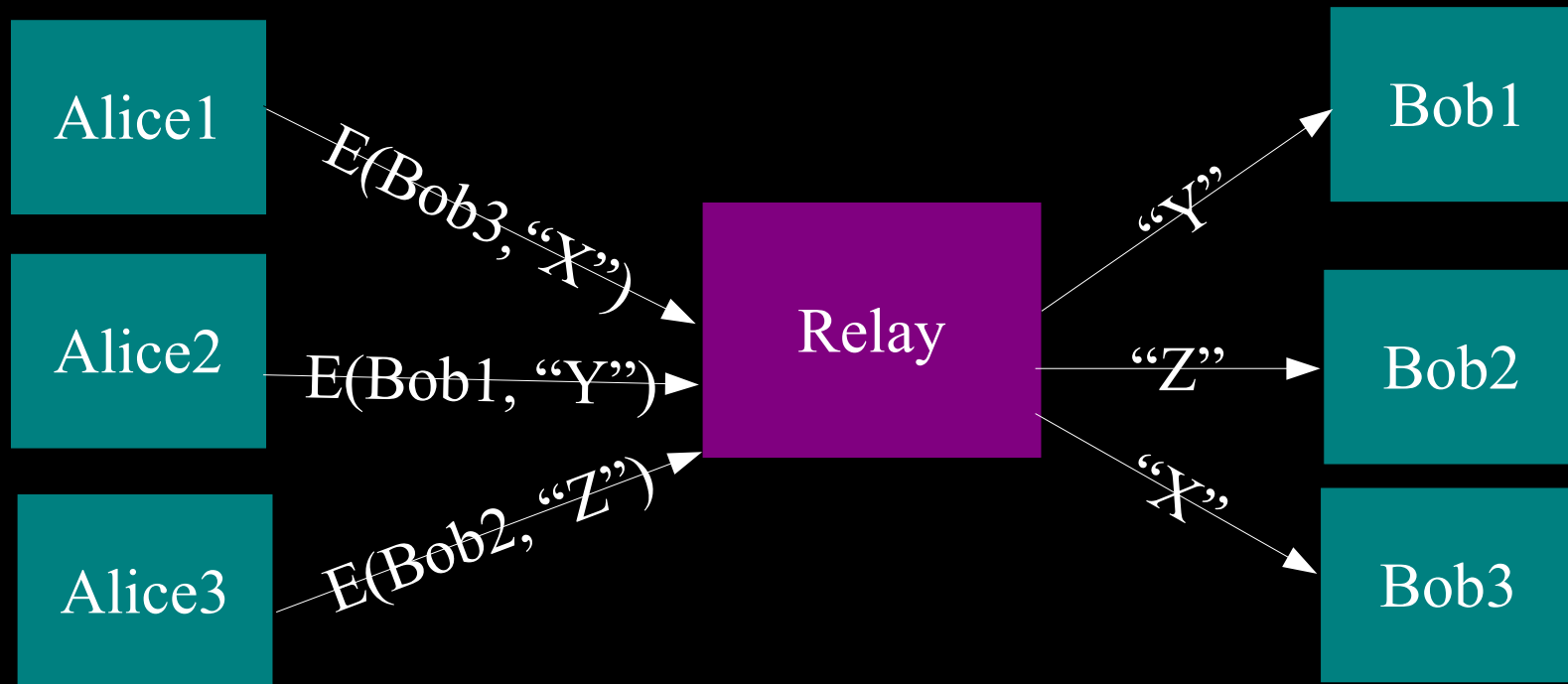
“It's privacy!”



# Anonymity serves different interests for different user groups.

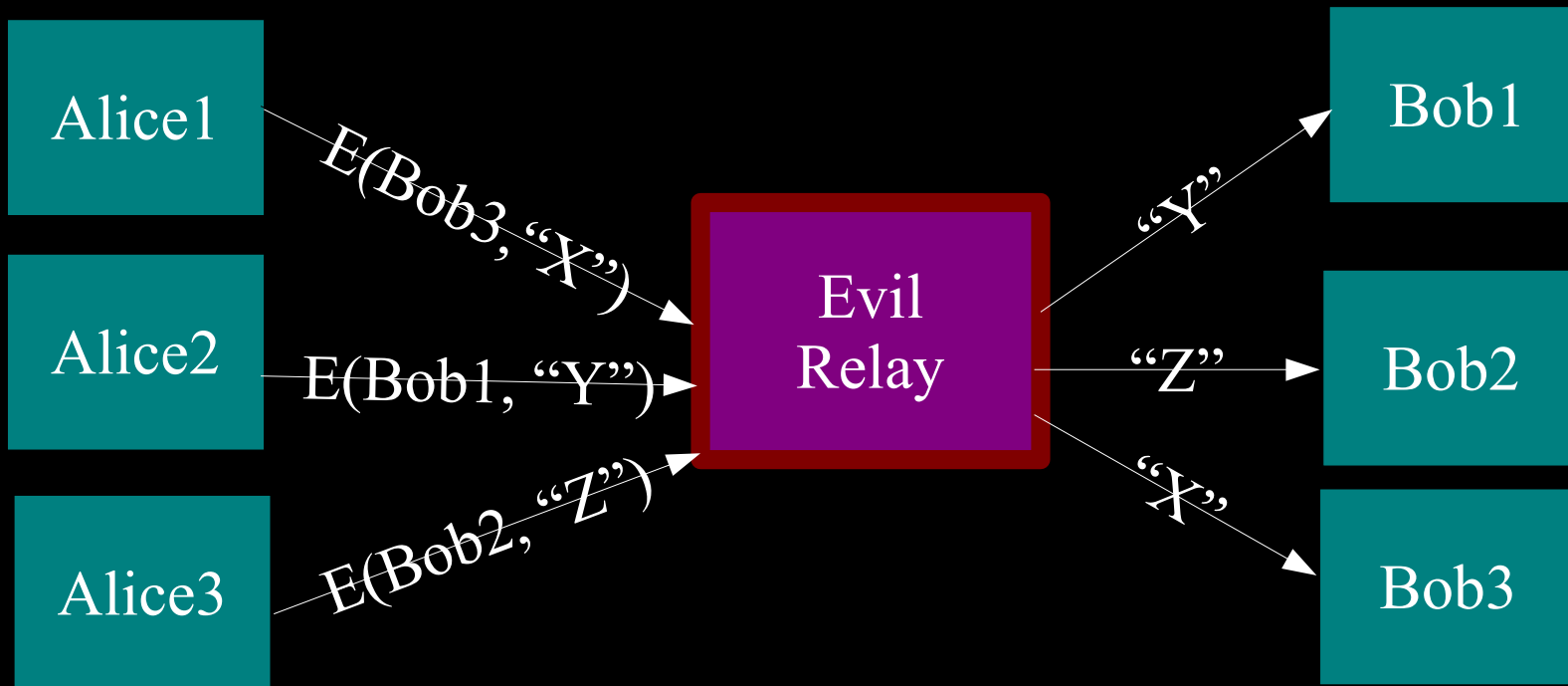


# The simplest designs use a single relay to hide connections.



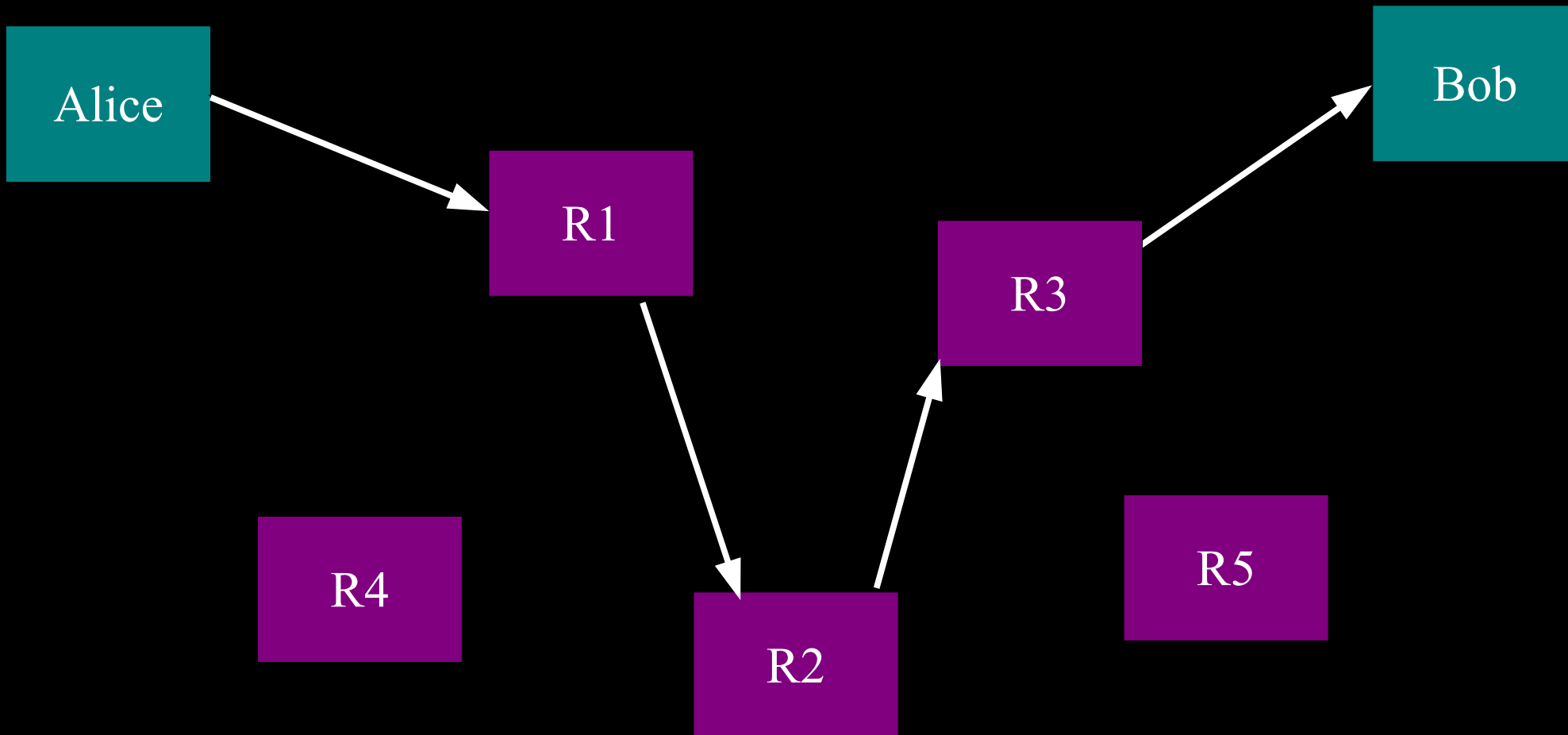
(example: some commercial proxy providers)

**But a single relay (or eavesdropper!)  
is a single point of failure.**

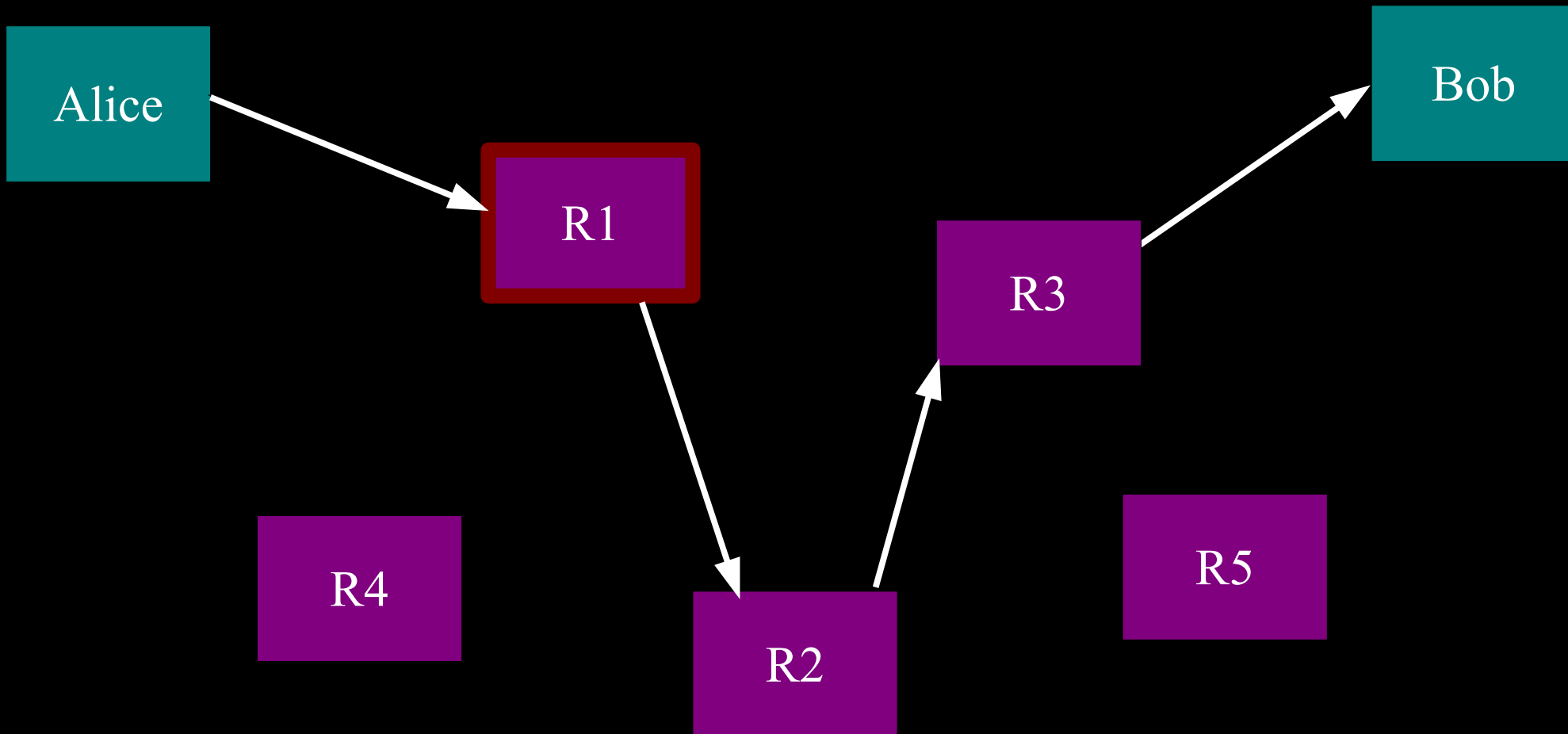




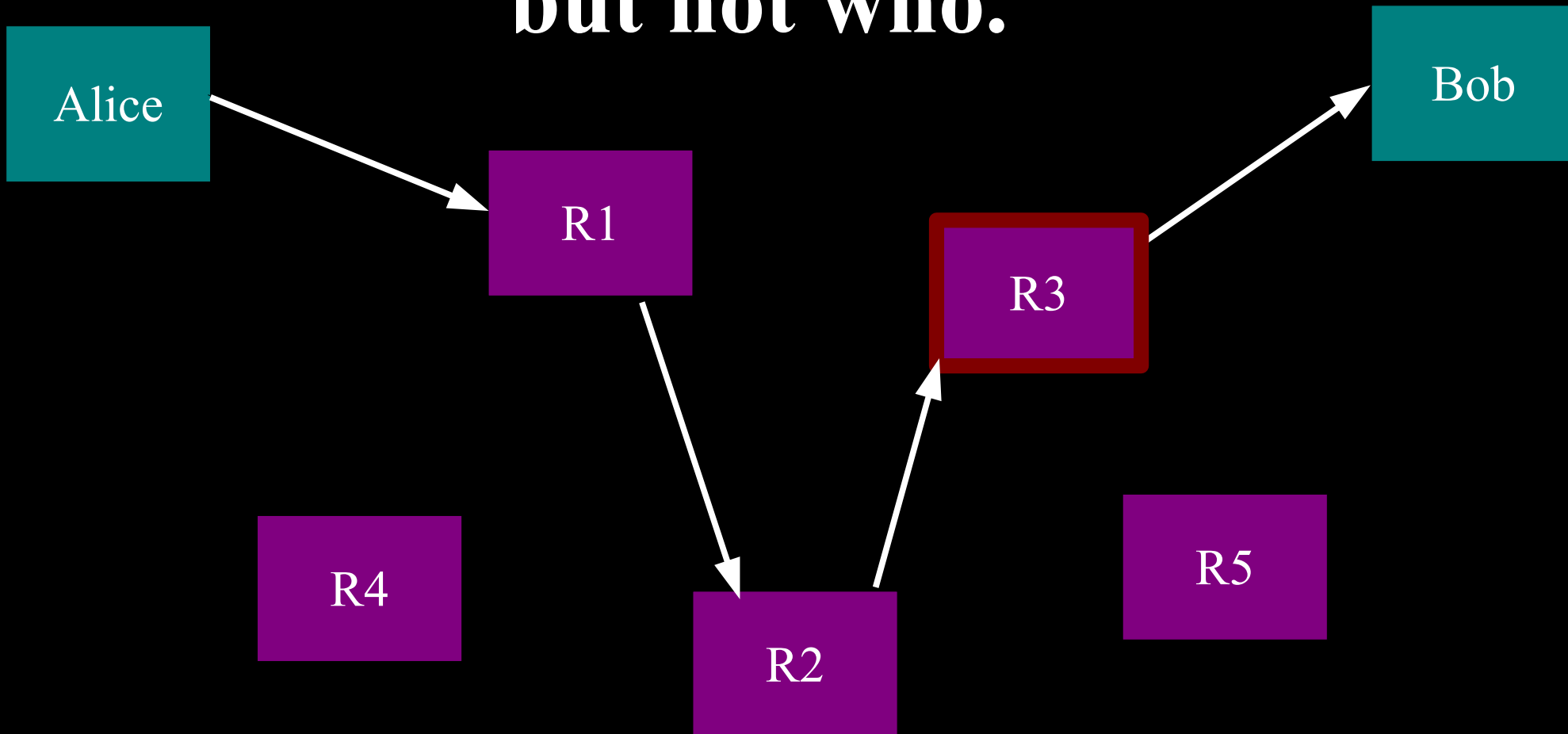
**So, add multiple relays so that no single one can betray Alice.**



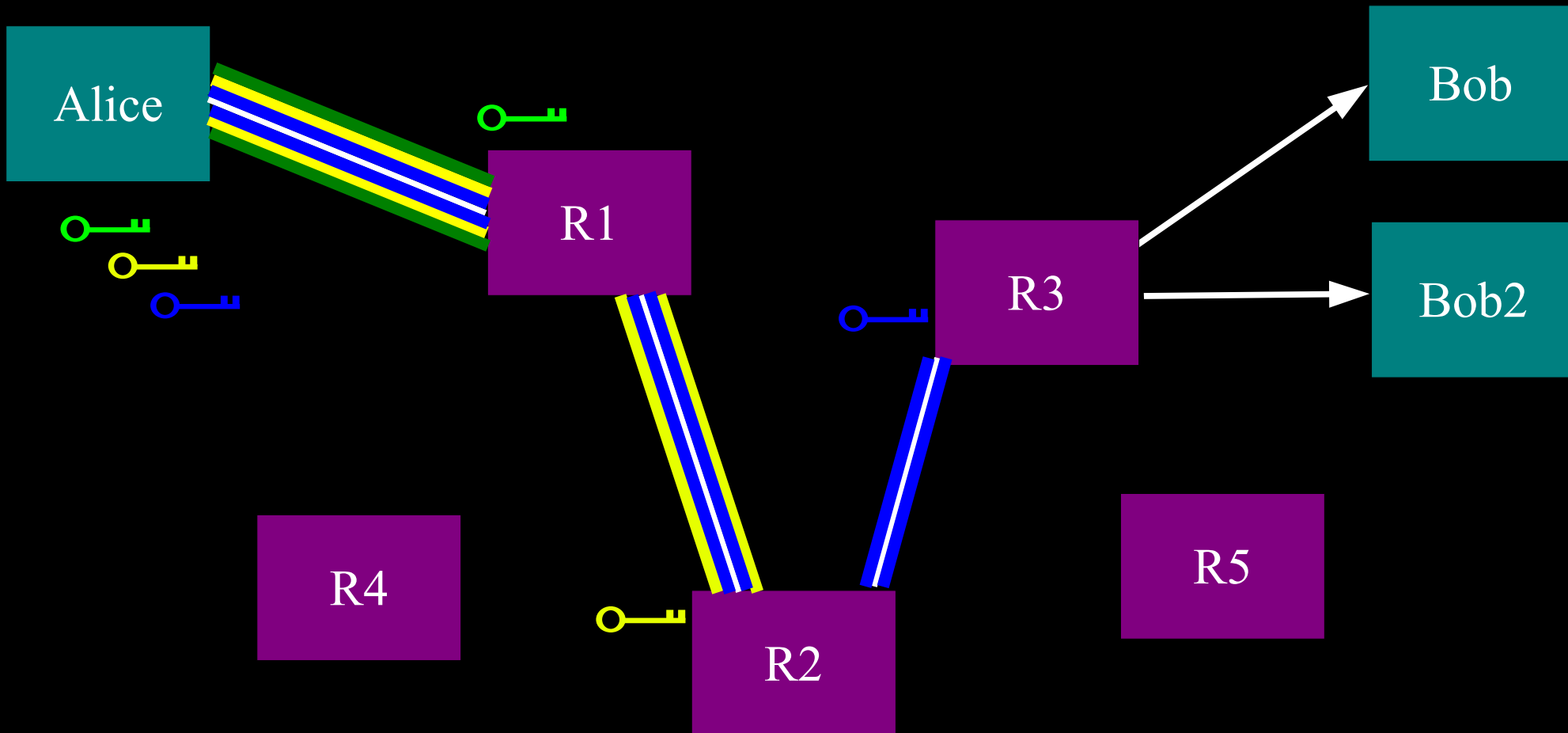
**A corrupt first hop can tell that Alice is talking, but not to whom.**



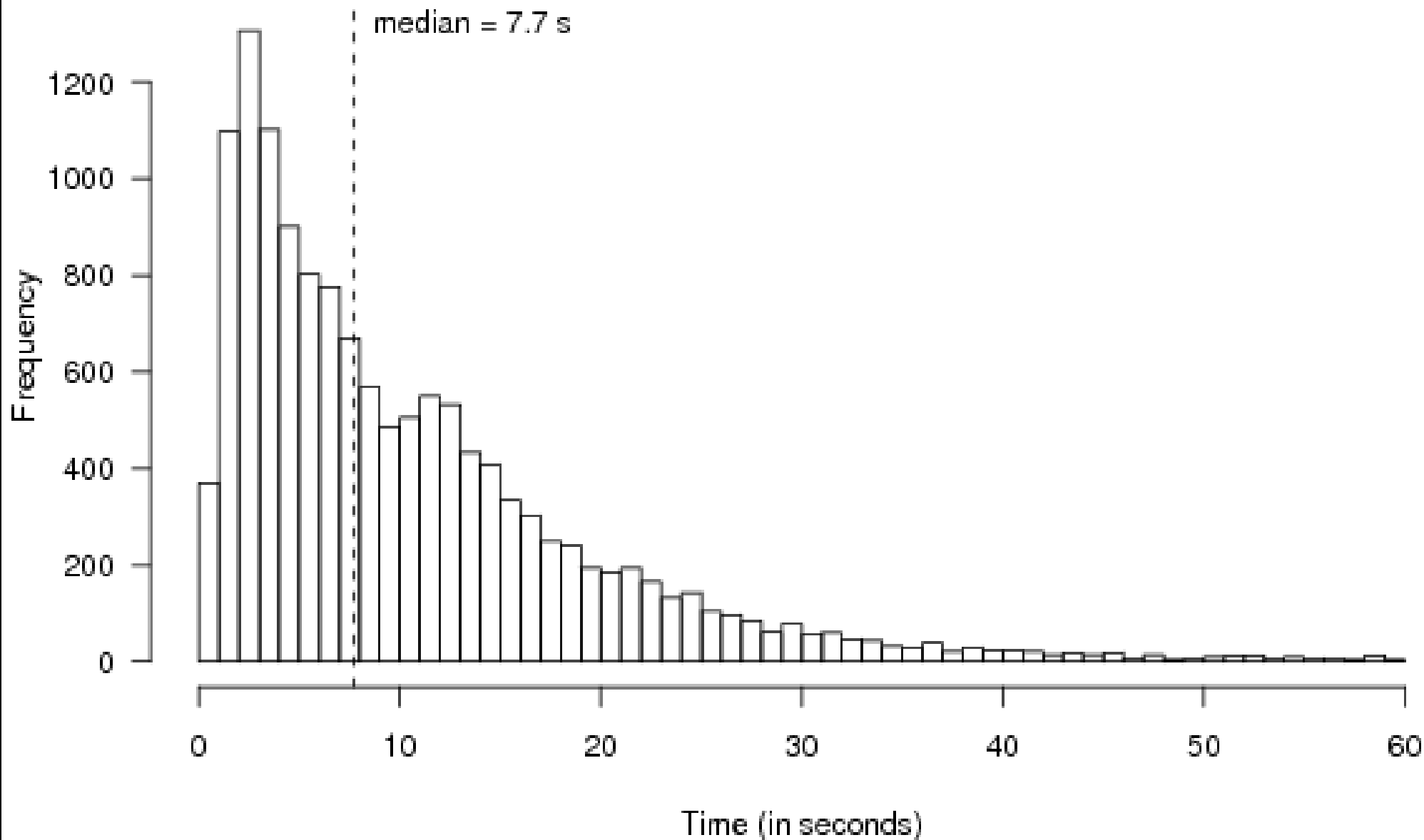
**A corrupt final hop can tell that somebody is talking to Bob, but not who.**



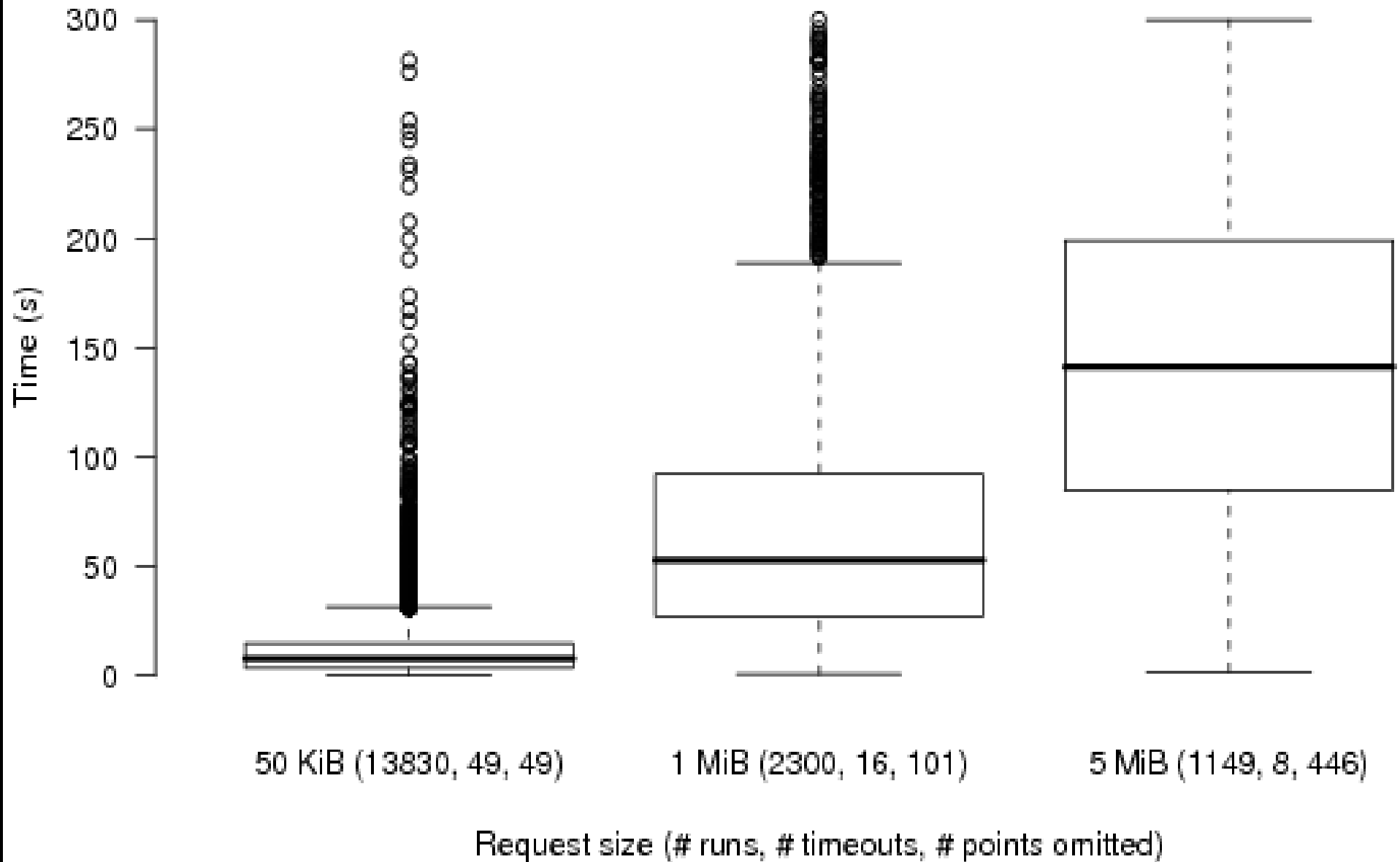
**Alice makes a session key with R1  
...And then tunnels to R2...and to R3**



### Download times for 50 KiB files



## Time to complete request



# Six performance problems

- Tor's congestion/flow control is not good
- Some users bulk-transfer over Tor
- Not enough capacity (run a relay!)
- Load balancing isn't right
- Not just high latency, but high variability
- High directory downloading overhead

# Six performance problems

- *Tor's congestion/flow control is not good*
- Some users bulk-transfer over Tor
- Not enough capacity (run a relay!)
- Load balancing isn't right
- Not just high latency, but high variability
- High directory downloading overhead



# TCP backoff slows down every circuit at once (1)

- To multiplex many circuits over a given TCP connection
- The only trick TCP has to slow one down is to slow them all down
- Especially bad on asymmetric bandwidth links (cablemodem, DSL, ...)

# TCP backoff slows down every circuit at once (2)

- The solution: switch to a datagram protocol (e.g. UDP) and layer end-to-end flow control on top of it.
- Needs a secure maintained free-software portable user-space TCP stack? Yuck.
- Maybe other datagram protocols have better congestion control. Delay-based backoff rather than drop-based?

# Circuit window sizes too big?

- Tor does flow control with end-to-end “circ window” plus “sendme” ack cells
- Fixed-size window of 1000 cells (512KB)
- Cutting the window size to 100 reduces buffer sizes (and queues), but increases roundtrips

# Six performance problems

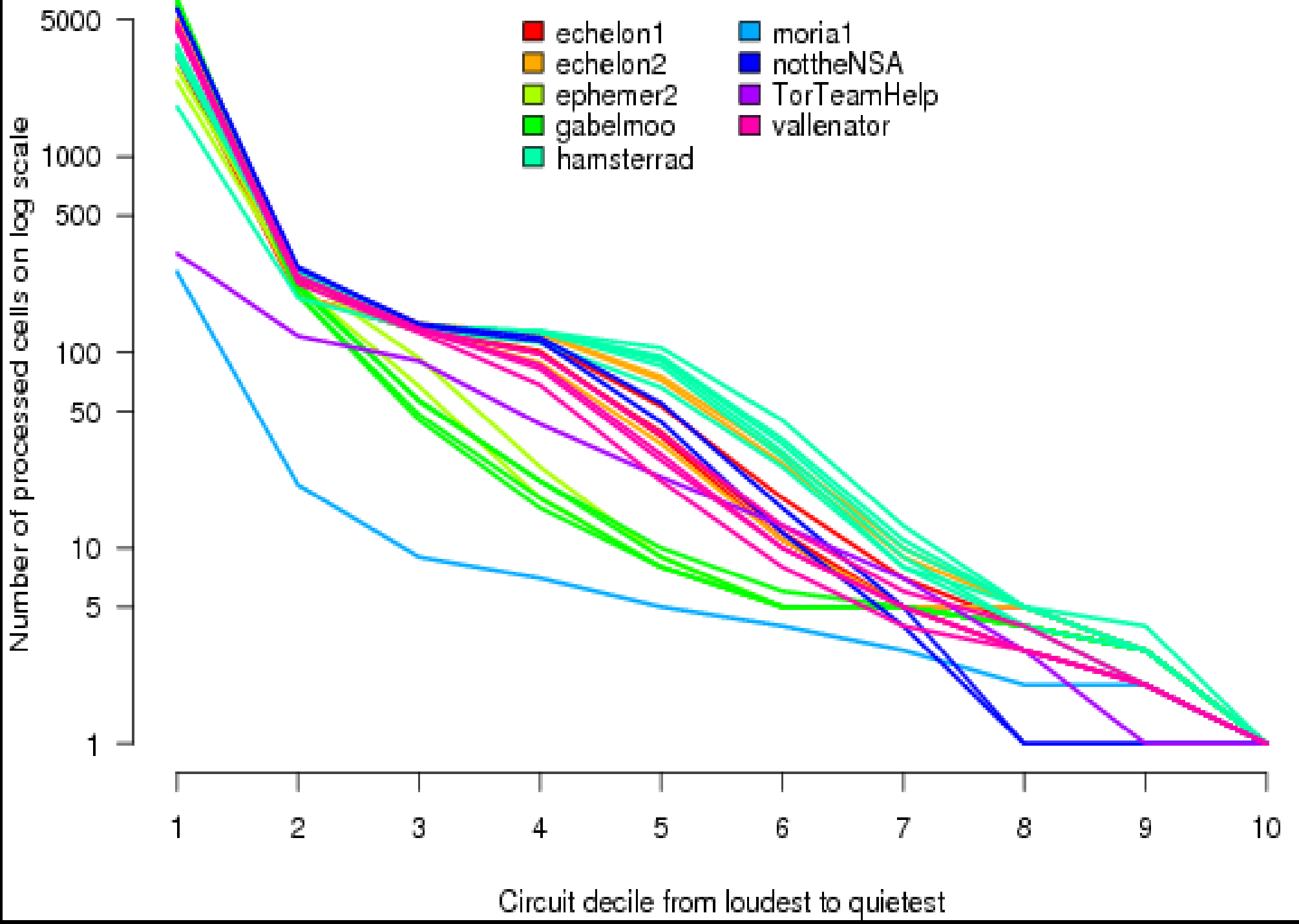
- Tor's congestion/flow control is not good
- *Some users bulk-transfer over Tor*
- Not enough capacity (run a relay!)
- Load balancing isn't right
- Not just high latency, but high variability
- High directory downloading overhead

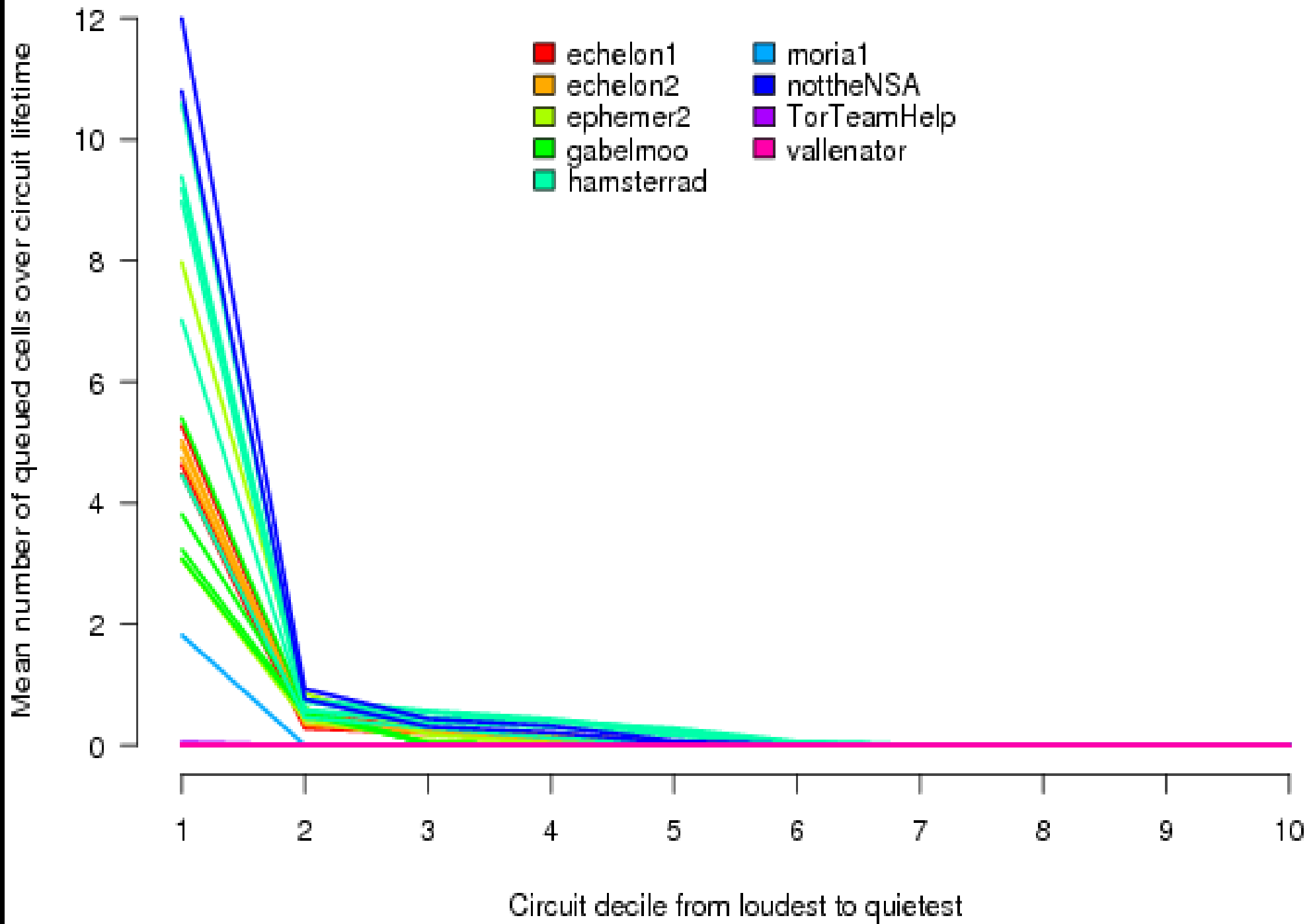
# Lessons from economics

- Increase in supply (network capacity) means increase in demand (users)
- We used to think there would be an equilibrium
- But file-sharing users have a different tolerance for latency than web browsing users

# Squeeze over-active circuits (1)

- Right now we round-robin among all “active” circuits when choosing next cell
- Most relays rate-limit: they'll only deliver a certain number of cells per second
- So circuits that are always active end up sending more cells.







# Squeeze over-active circuits (2)

- So we should pick from the really loud circuits less often.
- But using what algorithm?
- And how do we know whether we'll actually make it better?

# Throttle bandwidth in client

- Not really a stable solution, since users could “fix” their client
- But can't do it at the relay, since the relays would need to coordinate what they see
- Throttling bandwidth at the client can actually make you more secure, too!  
Cf. the paper that Columbia is working on

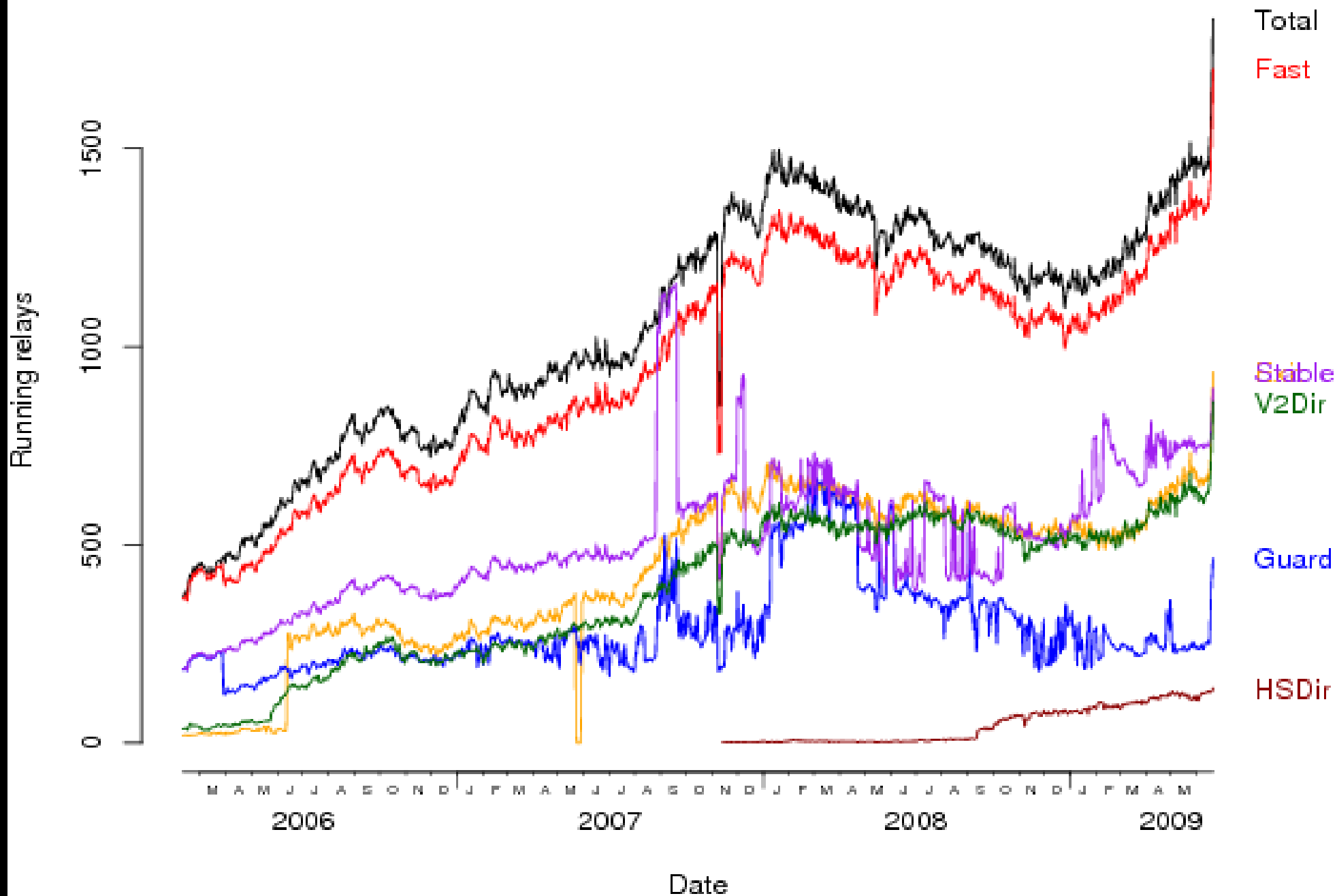
# Six performance problems

- Tor's congestion/flow control is not good
- Some users bulk-transfer over Tor
- *Not enough capacity (run a relay!)*
- Load balancing isn't right
- Not just high latency, but high variability
- High directory downloading overhead

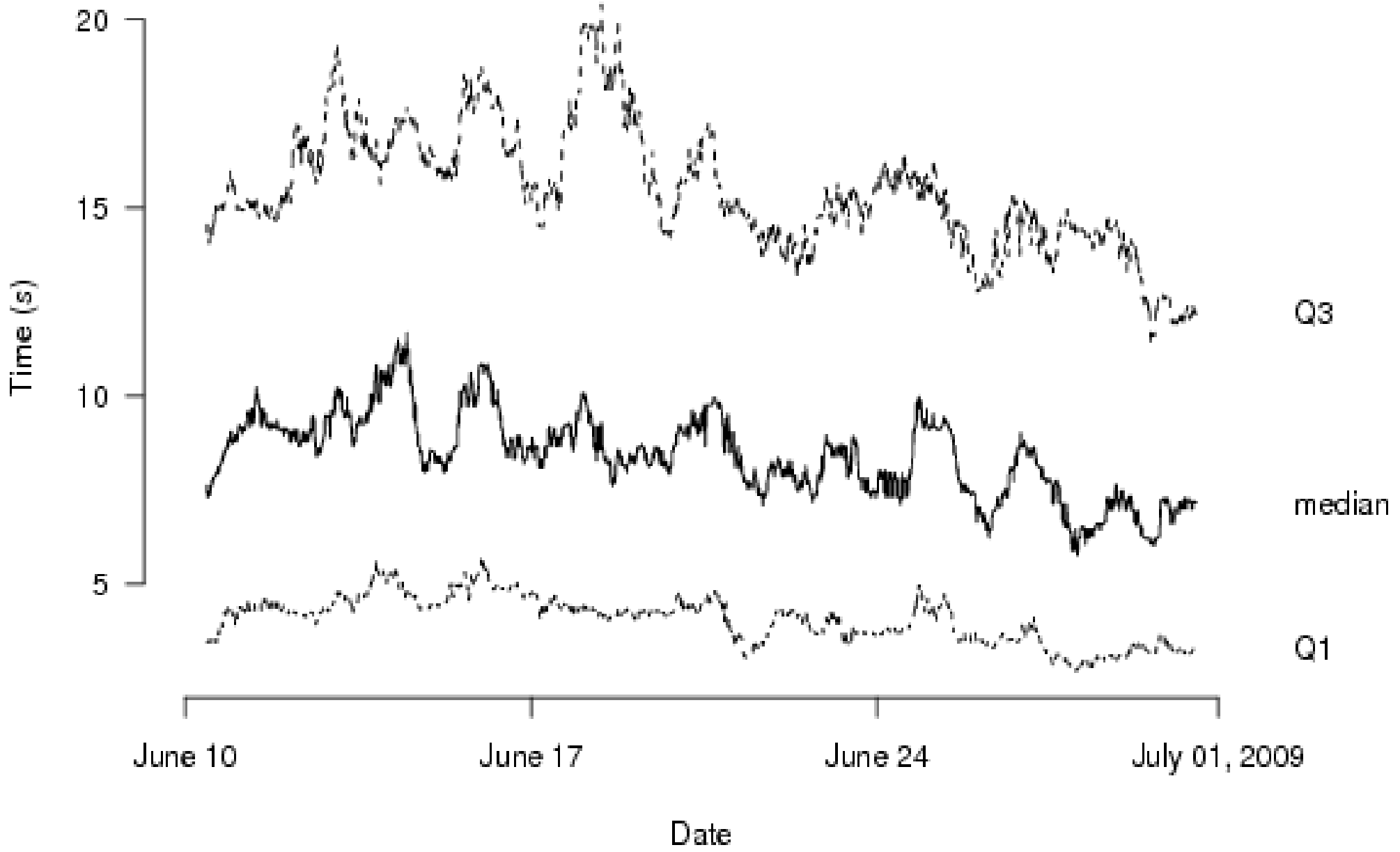
# Why is capacity only #3?

- If congestion control continues to be poor, getting more relays won't solve that
- Won't bulk-transfer users expand to fill our new capacity?
- Remember our economic argument

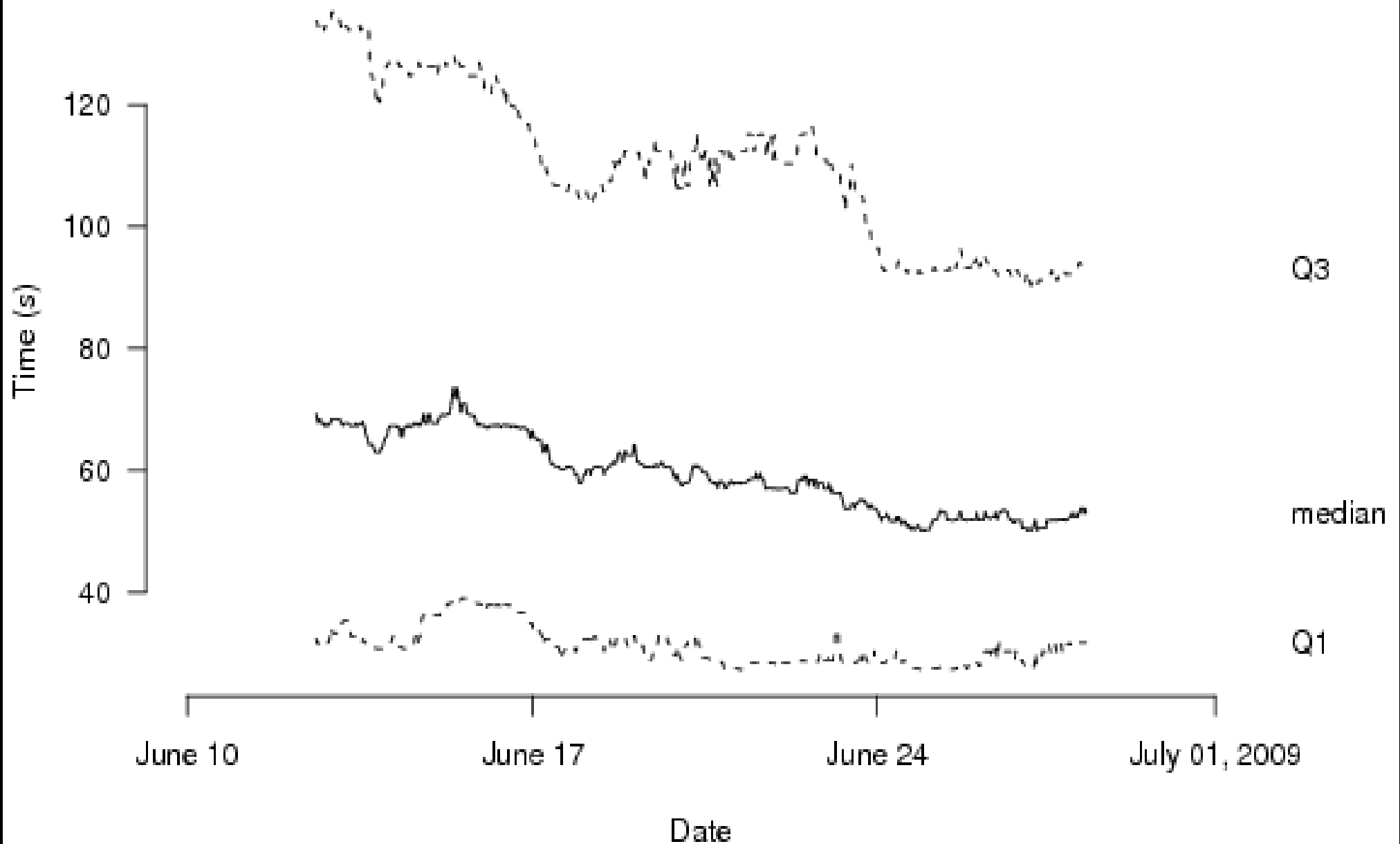
# Relay statuses



Time to complete request (50 KiB, 5950 completed runs, 16 timeouts)



Time to complete request (1 MiB, 993 completed runs, 4 timeouts)

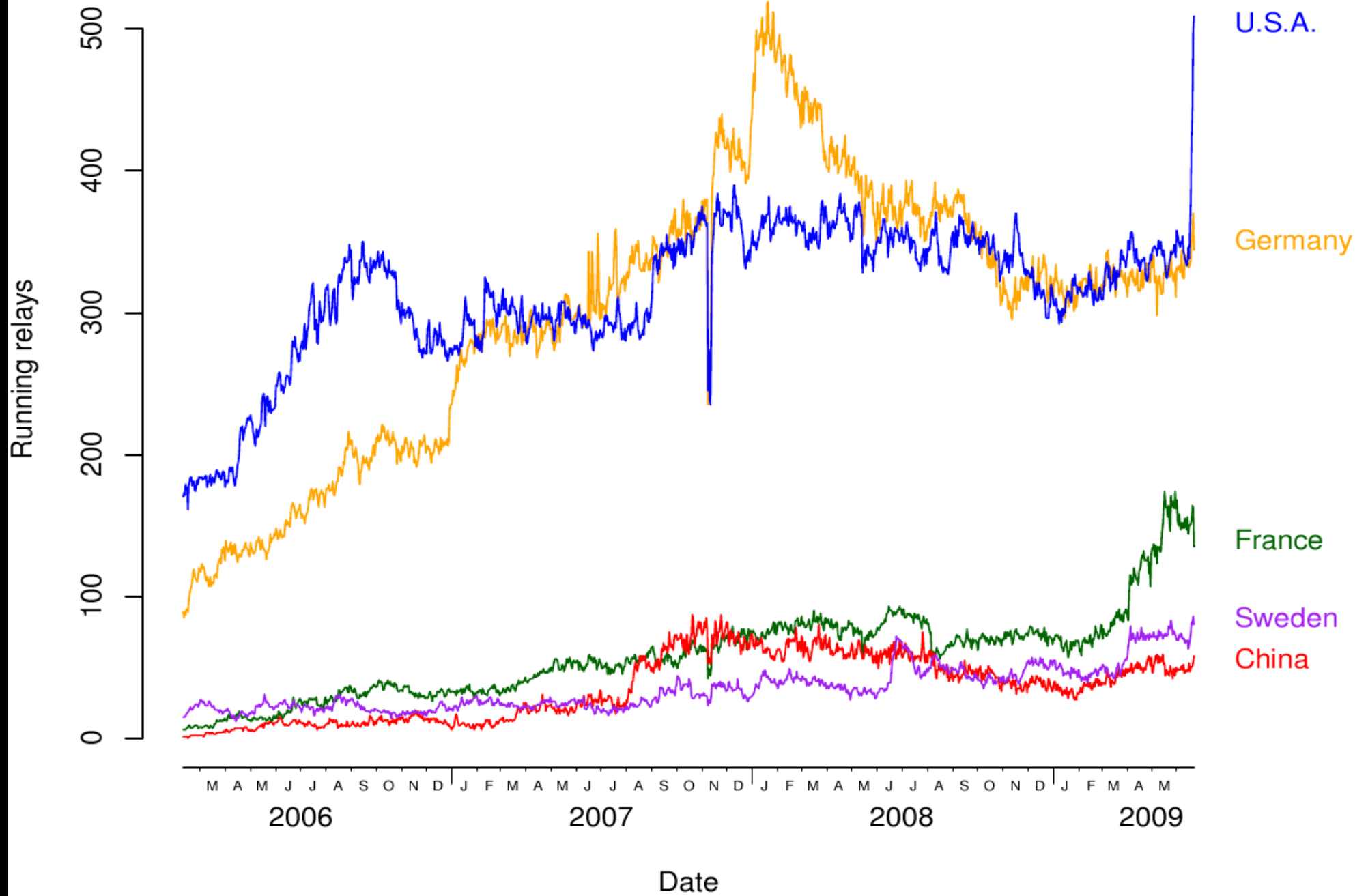


# Relay advocacy

- Jake and I keep doing talks and trainings all over the world
- Need better support for relay operators
  - Mailing list just for them?
  - “Tor weather” cgi to mail them when their relay goes down



# Relay locations



# Incentive mechanisms

- Gold-star reputation design: be a relay, get rewarded with better performance.
- Micropayment approaches
- But: intersection attacks on lists of running relays

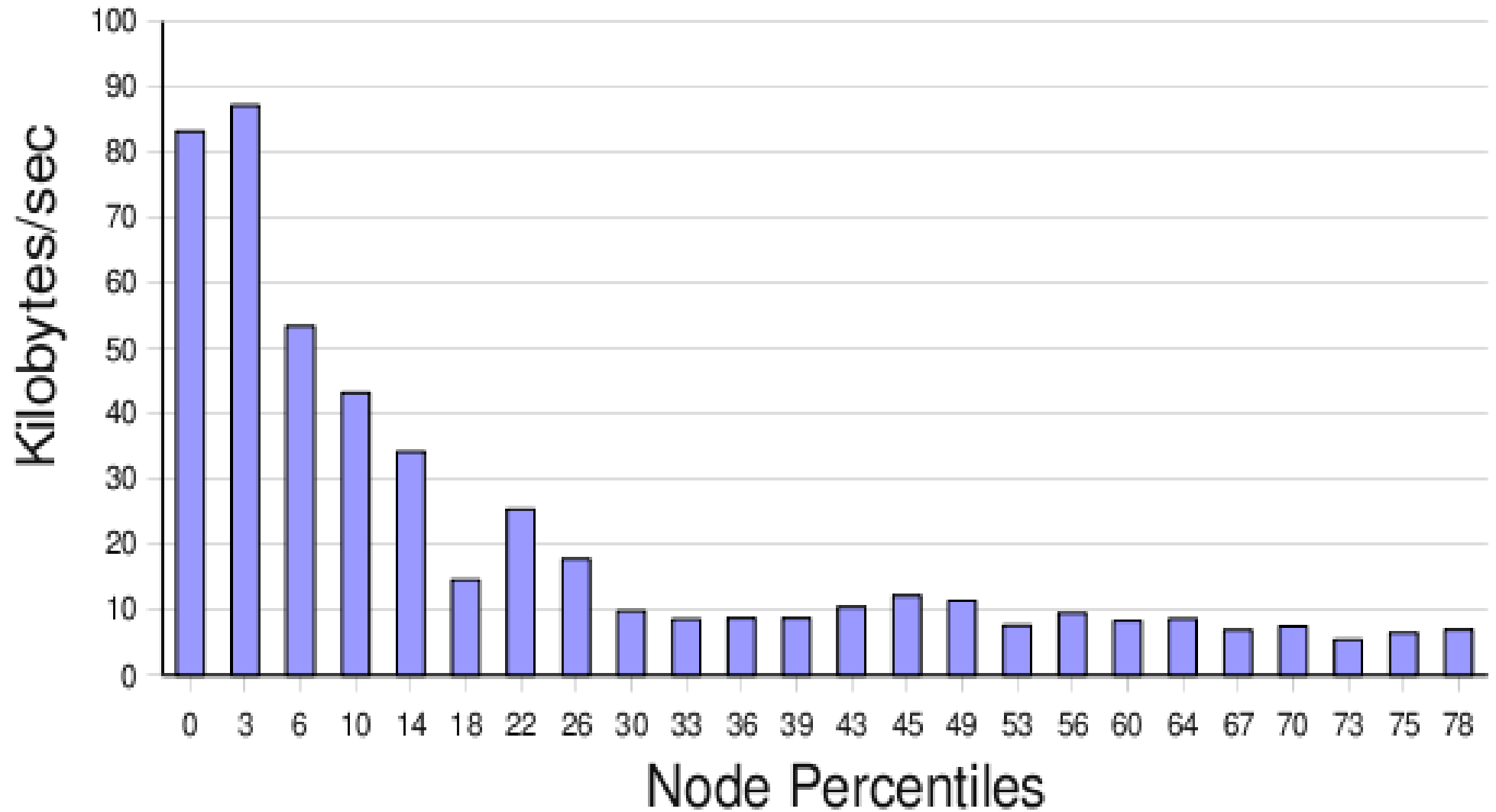
# Everybody-a-relay

- Need to support fast Tor windows relays (Nick has spent the past months hacking libevent / openssl)
- Automatically configure rate limiting?
- Need a directory design that scales
- Anonymity risks from letting the attacker relay traffic through you

# Six performance problems

- Tor's congestion/flow control is not good
- Some users bulk-transfer over Tor
- Not enough capacity (run a relay!)
- *Load balancing isn't right*
- Not just high latency, but high variability
- High directory downloading overhead

# Avg Stream Bandwidth in 2009



# Torflow: better bandwidth weights via measuring

- Bandwidth self-measuring not so good
- And we had to cap it at 10MB/s to resist cheaters
- Now we actively measure, and put the results in the consensus for clients
- Still a tradeoff between optimal network use vs anonymity

# Old entry guards are overloaded

- The longer you're an entry guard, the more clients you accumulate
- Now clients expire each guard after a month
- (This issue also means that brand new entry guards have no users, so aren't used efficiently)

# What about one-hop paths?

- It used to be a bad idea because it would screw up load balancing. Not so bad now.
- They're clearly way worse for anonymity.
- If exits are scarce, would it actually help?
- The main stumbling block is exit relay exposure: they'd become juicy targets, since no more guaranteed distributed trust



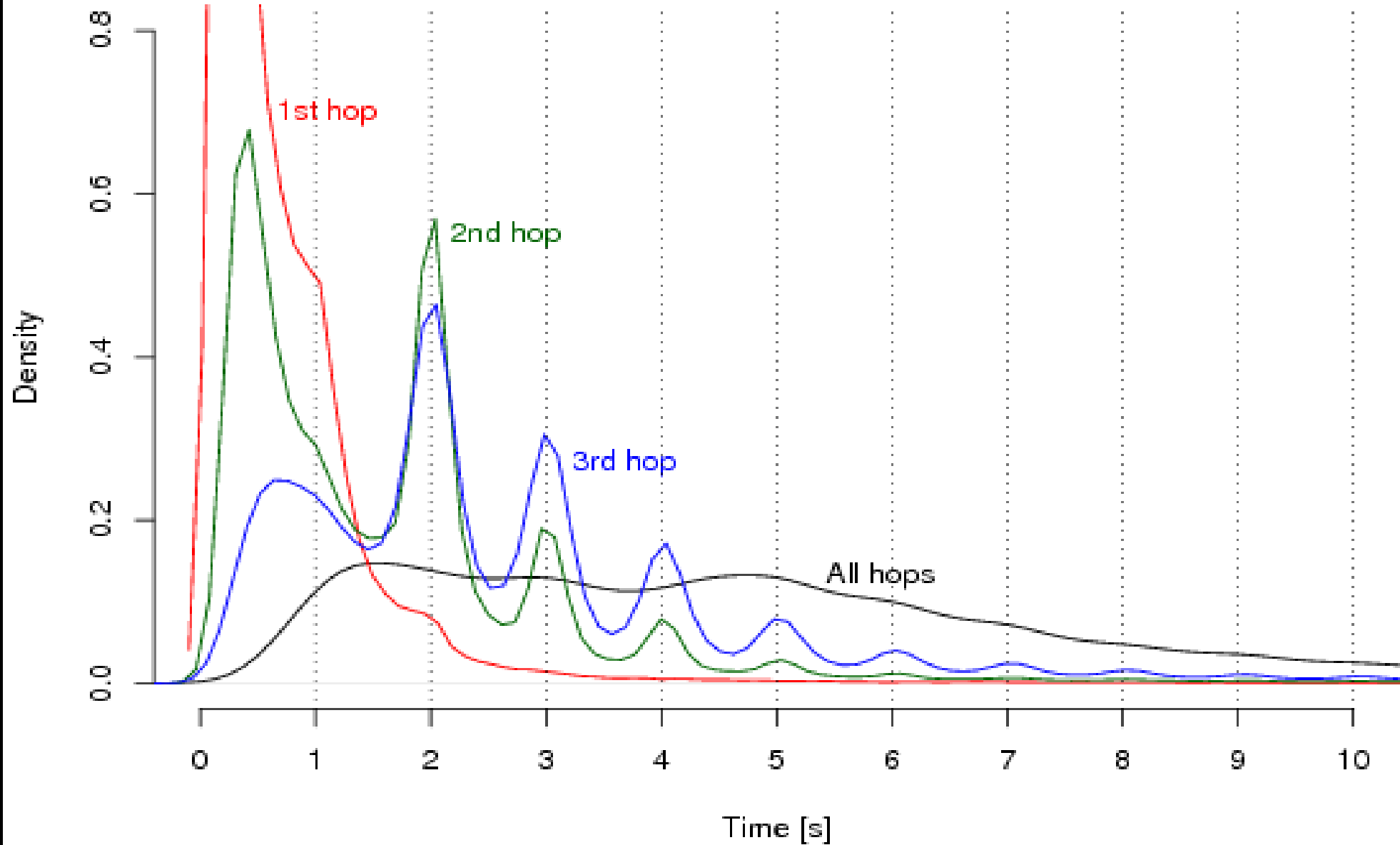
# Six performance problems

- Tor's congestion/flow control is not good
- Some users bulk-transfer over Tor
- Not enough capacity (run a relay!)
- Load balancing isn't right
- *Not just high latency, but high variability*
- High directory downloading overhead

# Per-second rate limiting

- Tor uses a token bucket for its rate limiting. It refills the bucket each second.
- Now that relays are overloaded, that means a burst of traffic at the beginning of each second, and then silence.

### Circuit extension time



# Adaptive circuit build timeouts

- Some circuits finish building in a few seconds. Some take 15-20 seconds.
- Circuits that build slowly also have bad performance. We should discard them.
- We can't just lower the timeouts: folks in Zimbabwe would never finish a circuit
- Need to measure build times at the client and dynamically adapt the timeouts

# Same thing for stream timeouts?

- Right now our stream timeouts are hard-coded at 10sec for the first two attempts, 15sec for later attempts.
- This is way too low for people on modems in Iran.
- So even if the user is really patient, their Tor client isn't.

# Six performance problems

- Tor's congestion/flow control is not good
- Some users bulk-transfer over Tor
- Not enough capacity (run a relay!)
- Load balancing isn't right
- Not just high latency, but high variability
- *High directory downloading overhead*

# Clients need to learn about available relays

- The quicker the client learns, the more use we get from short-term relays
- Clients need to share the same view of the network to prevent partitioning attacks
- We want it to scale to many thousands of relays

# Scaling directory info

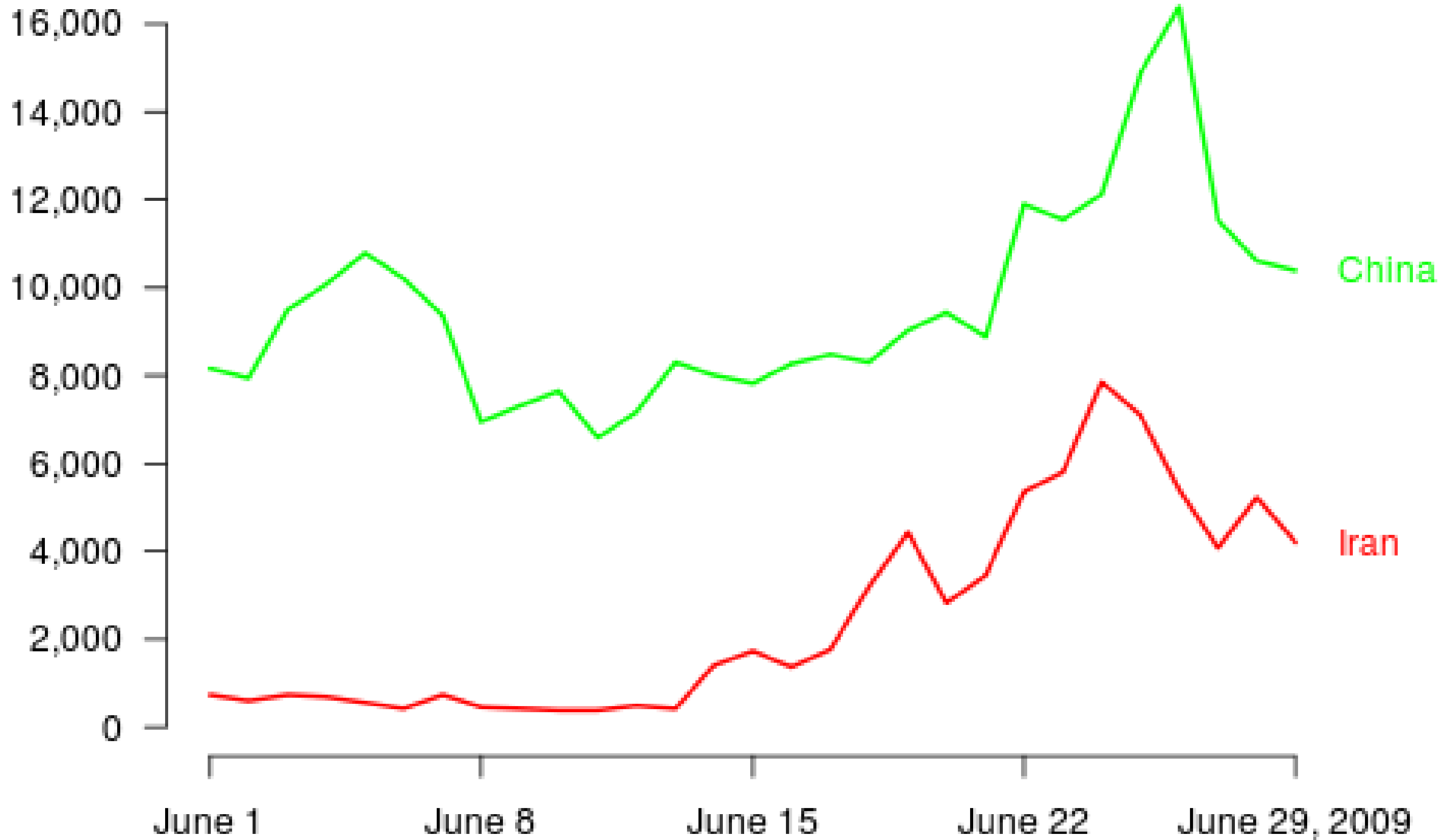
- V1 directory design: big list of descriptors
- V3 directory design: networkstatus consensus, plus individual descriptors
- Microdescriptor design: networkstatus consensus, plus mostly static microdescs
- Consensus diffs?



# Last thoughts

- How do we decide whether a given design change will actually help?
- Tor network simulator sure would be nice
- Doing measurements is also a good start
  - We've got data!
- What about anonymity implications of our changes?

## New or returning Tor clients per day



<https://torproject.org>