Peer-to-Peer Systems and Security

Anonymity

Christian Grothoff

Technische Universität München

May 22, 2014

"The problem with losing your anonymity is that you can never go back." –Marla Maples

Motivation

Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?

Motivation

Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?

Eve cannot read the data Alice and Bob are sending, but:

- Eve knows that Alice and Bob are communicating.
- Eve knows the amount of data they are sending and can observe patterns.

$\Rightarrow$ Patterns may even allow Eve to figure out the data

How Much does TLS leak?

"We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack **identifies individual pages** in the same website with 89% accuracy, exposing personal details including **medical conditions**, financial and **legal affairs** and **sexual orientation**. We examine evaluation methodology and reveal accuracy variations as large as 18% caused by assumptions affecting caching and cookies." [3]

Anonymity Definitions

Merriam-Webster:

1. not named or identified: “an anonymous author”, “they wish to remain anonymous”
2. of unknown authorship or origin: “an anonymous tip”
3. lacking individuality, distinction, or recognizability: “the anonymous faces in the crowd”, “the gray anonymous streets” – William Styron

Anonymity Definitions

Andreas Pfitzmann et. al.:

> "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

Anonymity Definitions

Andreas Pfitzmann et. al.:

> "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

EFF:

> "Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all)."

Anonymity Definitions

Andreas Pfitzmann et. al.:

"Anonymity is the state of being not identifiable within a set of subjects, the anonymity set."

EFF:

"Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all)."

Mine:

A user's action is anonymous if the adversary cannot link the action to the user's identity

The user's identity

includes personally identifiable information, such as:

- real name
- fingerprint
- passport number
- IP address
- MAC address
- login name
- ...

Actions

include:

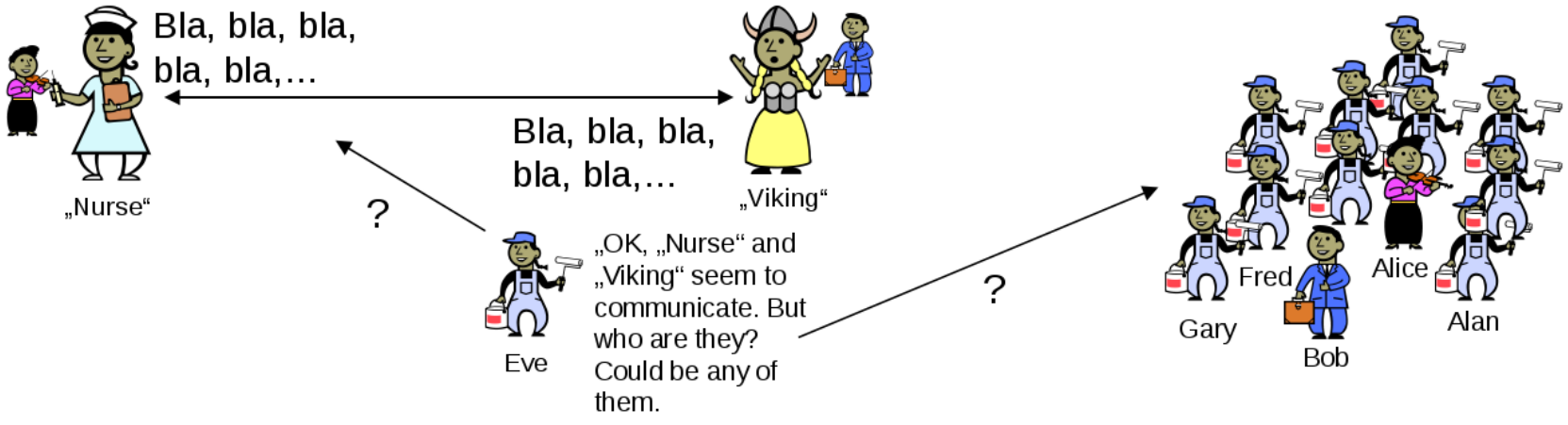
- Internet access
- speach
- participation in demonstration
- purchase in a store
- walking across the street
- ...

Anonymity: Terminology

- Sender Anonymity: The initiator of a message is anonymous. However, there may be a path back to the initiator.

- Receiver Anonymity: The receiver of a message is anonymous.

Pseudonymity



Pseudonymity

- A pseudonym is an identity for an entity in the system. It is a "false identity" and not the true identity of the holder of the pseudonym.
- Nobody, but (maybe) a trusted party may be able to link a pseudonym to the true identity of the holder of the pseudonym.
- A pseudonym can be tracked. We can observe its behaviour, but we do not learn who it is.

Evaluating Anonymity

How much anonymity does a given system provide?

- Number of known attacks?
- Lowest complexity of successful attacks?
- Information leaked through messages and maintenance procedures?
- Number of users?

Anonymity: Basics

- **Anonymity Set** is the set of suspects
- Attacker computes a **probability distribution** describing the likelyhood of each participant to be the responsible party.
- Anonymity is the stronger, the larger the anonymity set and the more evenly distributed the subjects within that set are.

Anonymity Metric: Anonymity Set Size

Let $\mathcal{U}$ be the attacker's probability distribution and $p_u = \mathcal{U}(u)$ describing the probability that user $u \in \Psi$ is responsible.

$$ASS := \sum_{\substack{u \in \Psi \\ p_u > 0}} 1 \tag{1}$$

Large Anonymity Sets

Examples of large anonymity sets:

- Any human

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human speaking English

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human speaking English
- Any human with phone access

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human speaking English
- Any human with phone access
- Any human with Internet access

Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human speaking English
- Any human with phone access
- Any human with Internet access
- Any human speaking English with Internet access

Anonymity Metric: Maximum Likelihood

Let $\mathcal{U}$ be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible.

$$ML := \max_{u \in \Psi} p_u \tag{2}$$

Anonymity Metric: Maximum Likelihood

- For successful criminal prosecution in the US, the law requires *ML close* to 1 ("beyond reasonable doubt")
- For successful civil prosecution in the US, the law requires $ML > \frac{1}{2}$ ("more likely than not")
- For a given anonymity set, the best anonymity is achieved if

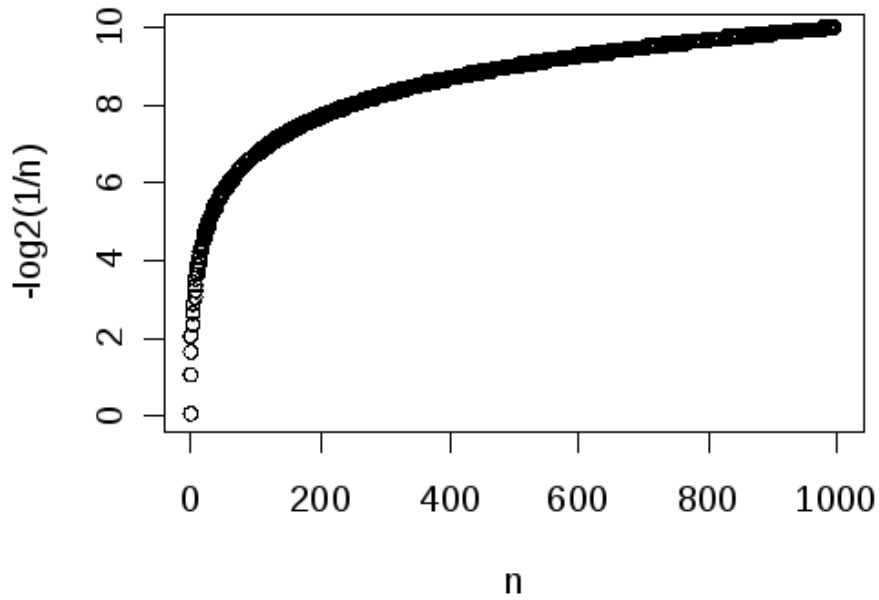
$$ML = \frac{1}{ASS} \tag{3}$$

Anonymity Metric: Entropy

Let $\mathcal{U}$ be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible. Define the effective size S of the anonymity distribution $\mathcal{U}$ to be:

$$S := -\sum_{u \in \Psi} p_u \log_2 p_u \tag{4}$$

where $p_u = \mathcal{U}(u)$.



Interpretation of Entropy

$$S = -\sum_{u \in \Psi} p_u \log_2 p_u \tag{5}$$

This is the *expected* number of bits of additional information that the attacker needs to definitely identify the user (with absolute certainty).

Entropy Calculation Example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S?

Entropy Calculation Example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S?

- For 101 nodes $H_{max} = 6.7$
-

$$S = -\frac{100 \cdot \log_2 0.001}{1000} - \frac{9 \cdot \log_2 0.9}{10} \tag{6}$$

$$\approx 0.9965 + 0.1368 \tag{7}$$

$$= 1.133... \tag{8}$$

Attacks to avoid

Hopeless situations include:

- All nodes collaborate against the victim
- All directly adjacent nodes collaborate
- All non-collaborating adjacent nodes are made unreachable from the victim
- The victim is required to prove his innocence

Economics & Anonymity

R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:

- Providing anonymity services has economic disincentives (DoS, legal liability)
- Anonymity requires introducing inefficiencies

$\Rightarrow$ Who pays for that?

Economics & Anonymity

R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:

- Providing anonymity services has economic disincentives (DoS, legal liability)
- Anonymity requires introducing inefficiencies

$\Rightarrow$ Who pays for that?

The anonymizing server that has the best reputation (performance, most traffic) is presumably compromised.

Anonymity: Dining Cryptographers

"Three cryptographers are sitting down to dinner. The waiter informs them that the bill will be paid anonymously. One of the cryptographers maybe paying for dinner, or it might be the NSA. The three cryptographers respect each other's right to make an anonymous payment, but they wonder if the NSA is paying." – David Chaum

PipeNet

Wei Dei suggested *PipeNet*:

- initiator knows receiver identity, but not vice-versa
- layered encryption, forwarding without delay
- constant traffic on each link to avoid observability

PipeNet

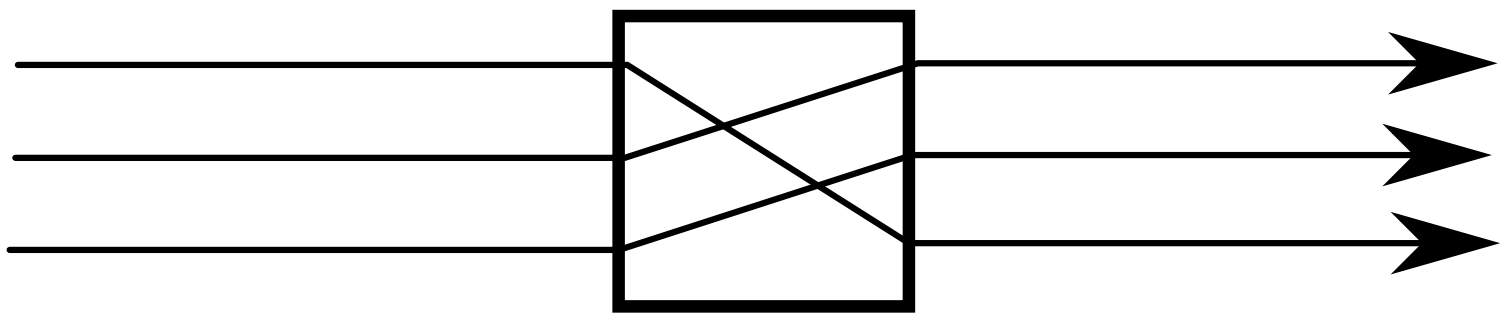
Wei Dei suggested *PipeNet*:

- initiator knows receiver identity, but not vice-versa
- layered encryption, forwarding without delay
- constant traffic on each link to avoid observability

Is this useful?

Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:

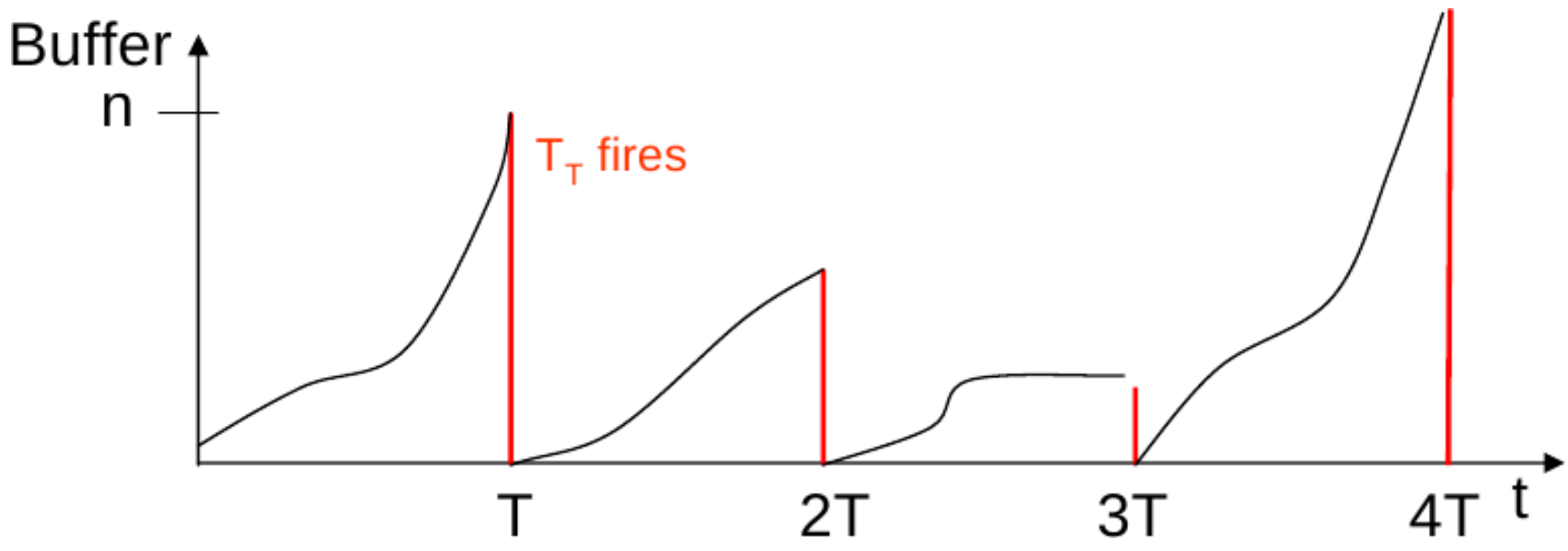


Mixing

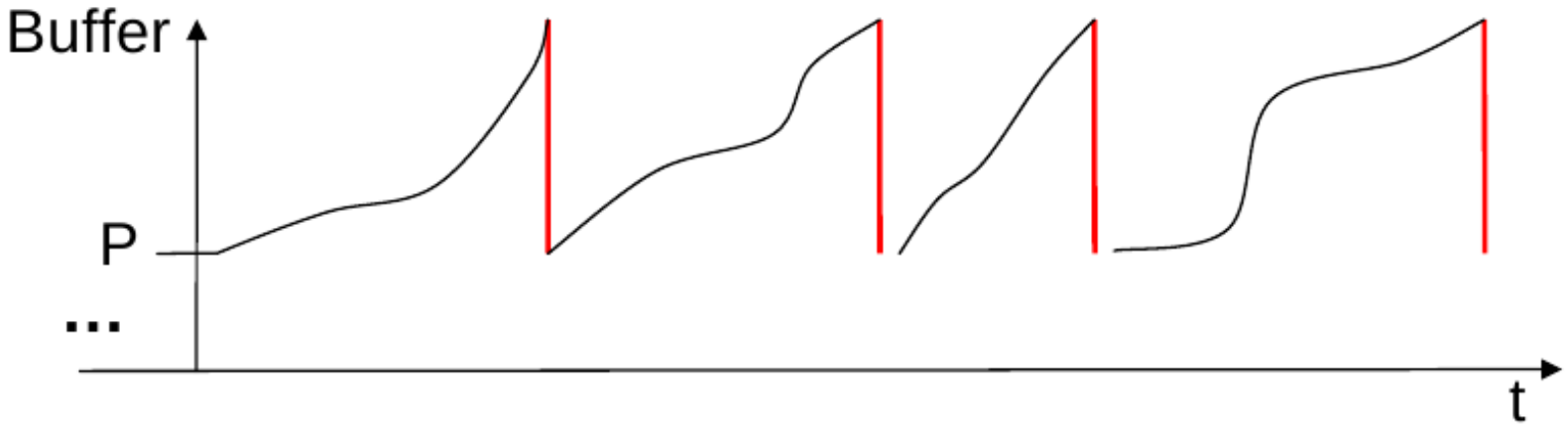
David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:

Threshold Mix

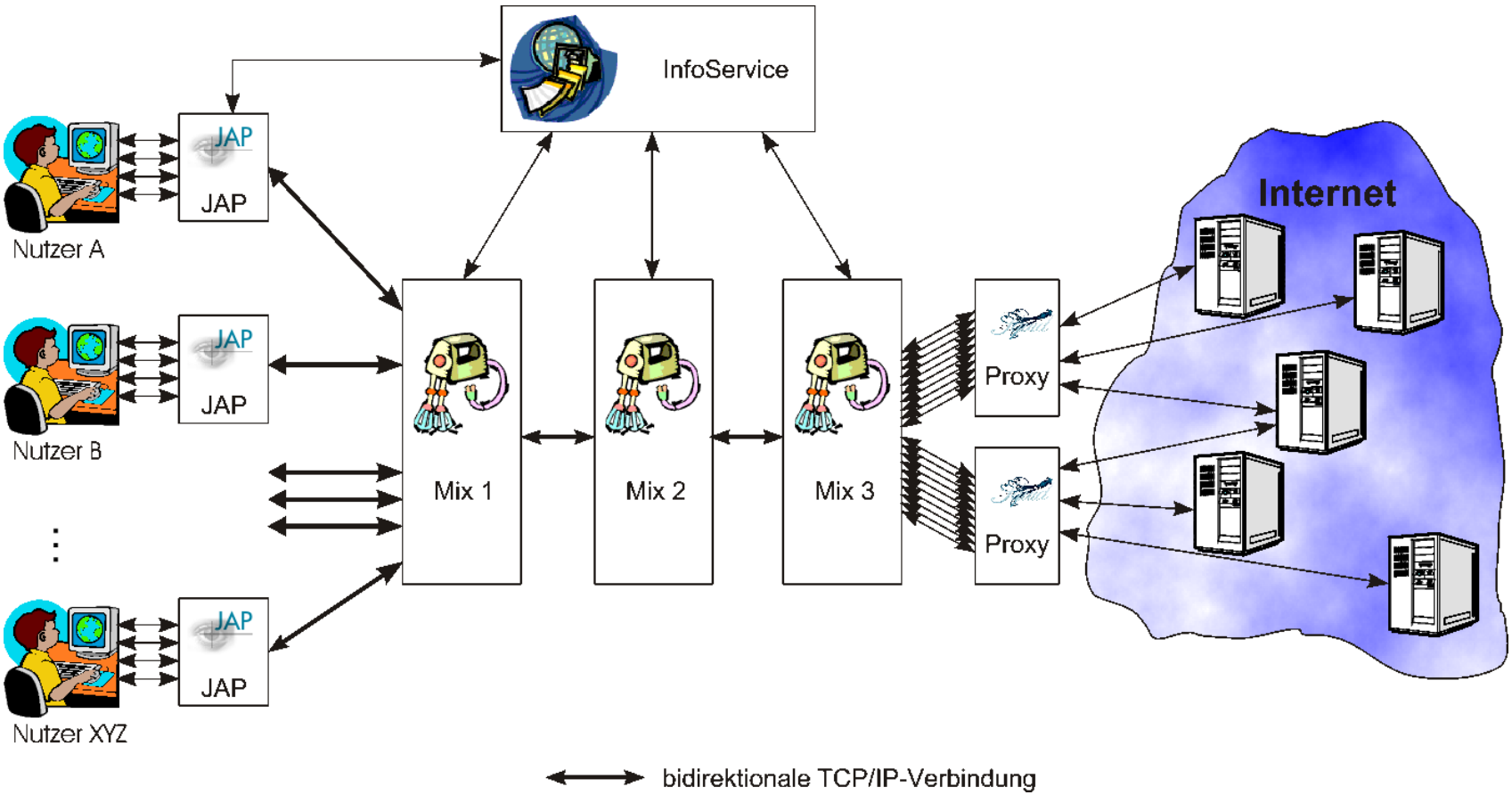
Timed Mix



Pool mix



JAP: Java Anonymizing Proxy[1]



[1] From Stefan Köpsell: "AnonDienst – Design und Implementierung", 2004

Crowds

M. Reiter and A. Rubin introduced Crowds [4]:

- primary application is web-surfing
- each member of the crowd chooses the next hop at random
- communication in the crowd is link-encrypted
- bi-directional communication (replies)
- efficiency and high scalability
- simple protocol

Crowds: non-goals

- no anonymity against local eavesdropper
- no responder anonymity
- no effort to defend against denial-of-service attacks (especially not against routers tampering with the indirected data)

Crowds: design

- node joins crowd (by signing up with central *blender* server), crowd forms one path with a single key for the entire path
- multiple, chained proxies, each proxy either exits or extends with probability $p_f > \frac{1}{2}$
- reply is routed back on the same path

Crowds: local eavesdropper

- there is no noise in the system

$\Rightarrow$ eavesdropper can see that a connection is initiated

- request is routed along static path with one session key

$\Rightarrow$ eavesdropper needs one node on the path for full exposure

Crowds: collaborating jondos

Suppose c out of n jondos are collaborating and p_f is the indirection probability. **Theorem 5.2:** If

$$n \geq \frac{p_f}{p_f - \frac{1}{2}} \cdot (c + 1)$$

the probability that the a collaborating jondo is the first node that the initiator connects to is lower than 50%.

Crowds: An attack[2]

The adversary may be able to deduce the initiator over time

- if an adversary controls one or more members of the crowd and
- if the protocol has multiple interactions between initiator and responder that can be correlated and that take different paths, since the initiator has a higher probability to be the sender of a query than all other nodes

[2] See also: M. Wright, M. Adler, B. Levine and C. Shields: *An Analysis of the Degradation of Anonymous Protocols*

Crowds: An attack[2]

The adversary may be able to deduce the initiator over time

- if an adversary controls one or more members of the crowd and
- if the protocol has multiple interactions between initiator and responder that can be correlated and that take different paths, since the initiator has a higher probability to be the sender of a query than all other nodes

Solution:

- try to use only one static path
- paths must change when new jondos join
- *solution:* new jondos must join in groups, controlled by the central registration server

[2]See also: M. Wright, M. Adler, B. Levine and C. Shields: *An Analysis of the Degradation of Anonymous Protocols*

Crowds: scalability

Since the amount of traffic a node receives depends only on p_f, not on the size n of the crowd, scalability is great.

The requirement that all paths must be re-formed whenever nodes join is *much* worse, especially since the anonymity of the system depends on large crowds.

Crowds: choosing p_f

- The network load on an individual jondo does not change at all if that jondo changes the parameter p_f.
- Since the last jondo on a path must decrypt, it is optimal for CPU load to choose $p_f = 0$.
- If a jondo chooses $p_f = 1$, this is optimal for the rest of the crowd (jondo becomes a proxy!).
- If the jondo then additionally follows the Crowd requirement to indirect its own requests, they are trivially detectable and the jondo itself is exposed.

Crowds: open problems

- exit nodes are fair game for legal actions
- no accounting to defend against abuse / DoS attacks
- lower indirection probability benefits only other nodes
  $\Rightarrow$ no possibility to trade efficiency for anonymity

Mixminion

G. Danezis, R. Dingledine, D. Hopwood and N. Mathewson describe Mixminion [2]:

- based on mixmailers (only application is E-mail)
- possibility to reply
- directory servers to evaluate participating remailers (reputation system)
- exit policies

Mixminion: key ideas

When a message traverses mixminion, each node must decrypt the message using its (ephemeral) private key.

The key idea behind the replies is splitting the path into two legs:

- the first half is chosen by the responder to hide the responder identity
- the second half was communicated by the receiver to hide the receiver identity
- a crossover-node in the middle is used to switch the headers specifying the path

Mixminion: replay?

Replay attacks were an issue in previous mixnet implementations.

- Mixes are vulnerable to replay attacks
- Mixminion: servers keep hash of previously processed messages until the server key is rotated

$\Rightarrow$ Bounded amount of state in the server, no possibility for replay attack due to key rotation

Mixminion: Directory Servers

- Inform users about servers
- Probe servers for reliability
- Allow a partitioning attack unless the user always queries all directory servers for everything

Mixminion: Nymservers

- Nymservers keep list of use-once reply blocks for a user
- Vulnerable to DoS attacks (deplete reply blocks)
- Nymservers could also store mail (use one reply block for many messages).

Mixminion: obvious problems

- no benefits for running a mixmailer for the operator
- quite a bit of public key cryptography
- trustworthiness of directory servers questionable
- servers must keep significant (but bounded) amount of state
- limited to E-mail (high latency)

Mixminion: open problems

- exit nodes are fair game for legal actions
- no accounting to defend against abuse / DoS attacks
- statistical correlation of entities communicating over time possible (observe participation)

$\Rightarrow$ bridging between an anonymous network and a traditional protocol is difficult

Tor

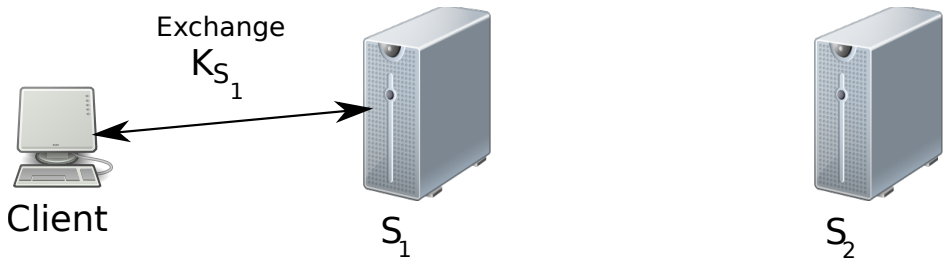
- Tor is a P2P network of **low-latency** mixes which are used to provide anonymous communication between parties on the Internet.
- Tor works for any TCP-based protocol
- TCP traffic enters the Tor network via a SOCKS proxy
- **Common usage:** client anonymity for web browsing

Onion Routing

- Multiple mix servers
- Path of mix servers chosen by initiator
- Chosen mix servers create "circuit"
  - Initiator contacts first server S_1, sets up symmetric key K_{S_1}
  - Then asks first server to connect to second server S_2; through this connection sets up symmetric key with second server K_{S_2}
  - ...
  - Repeat with server S_i until circuit of desired length n constructed

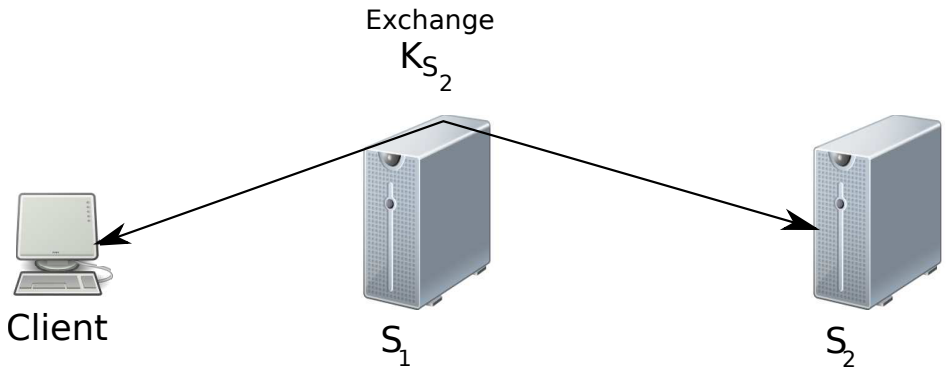
Onion Routing Example

- Client sets up symmetric key K_{S_1} with server S_1



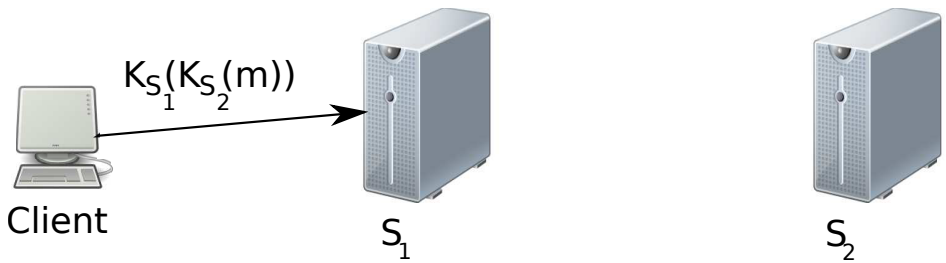
Onion Routing Example

- Via S_1 Client sets up symmetric key K_{S_2} with server S_2



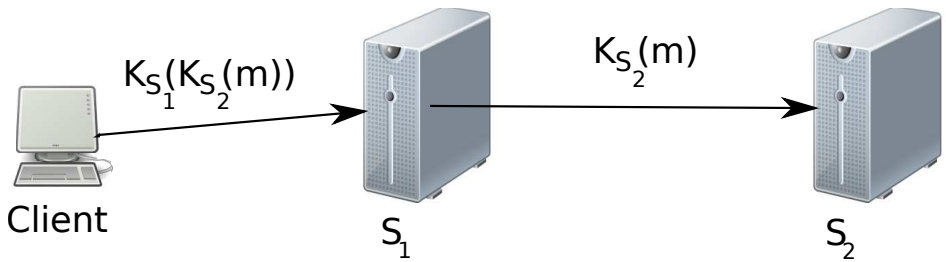
Onion Routing Example

- Client encrypts m as $K_{S_1}(K_{S_2}(m))$ and sends to S_1



Onion Routing Example

- S_1 decrypts, sends on to S_2, S_2 decrypts, revealing m

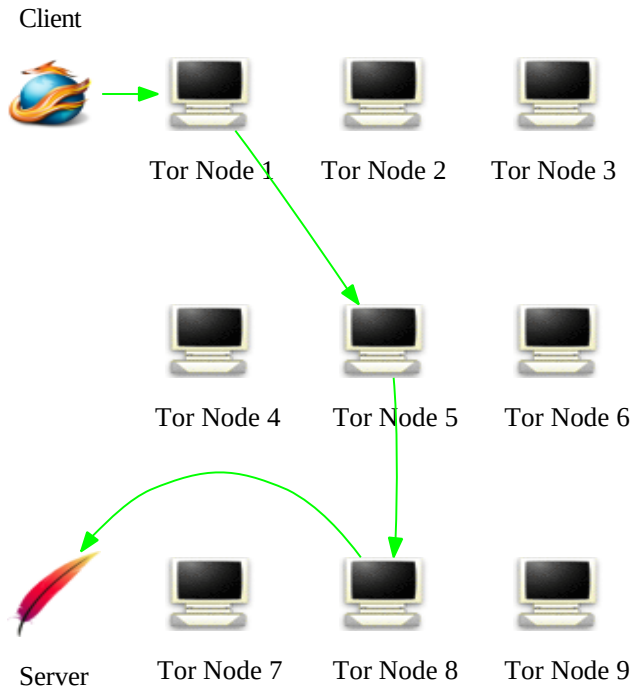


Tor - How it Works

- Low latency P2P Network of mix servers
- Designed for interactive traffic (https, ssh, etc.)
- "Directory Servers" store list of participating servers
  - Contact information, public keys, statistics
  - Directory servers are replicated for security
- Clients choose servers randomly with bias towards high BW/uptime
- Clients build long lived Onion routes "circuits" using these servers
- Circuits are bi-directional
- Circuits are of length three

Tor - How it Works - Example

- Example of Tor client circuit



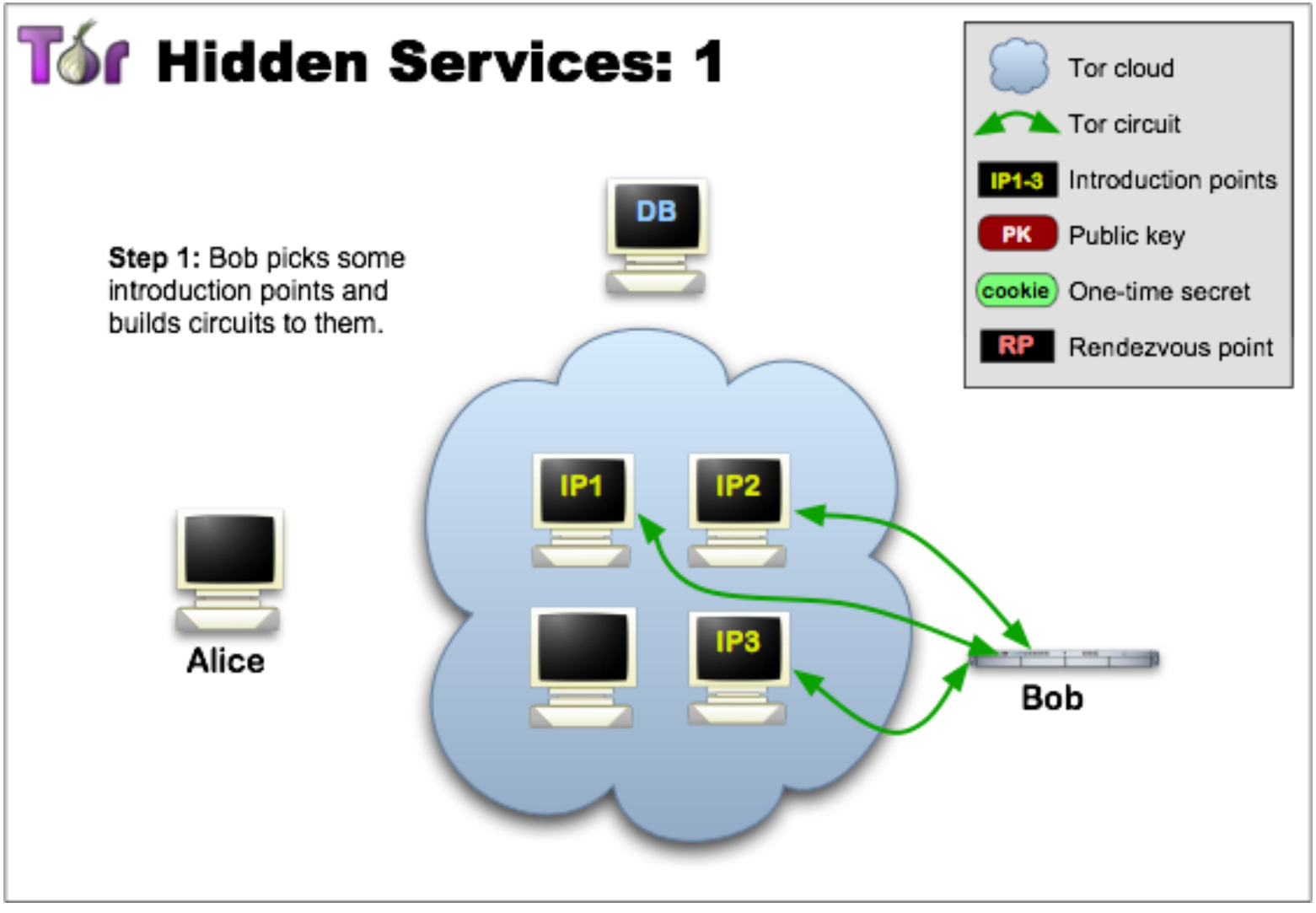
Tor - How it Works - Servers

- Servers are classified into three categories for usability, security and operator preference
- Entry nodes (aka guards) - chosen for first hop in circuit
  - Generally long lived "good" nodes
  - Small set chosen by client which are used for client lifetime (security)
- Middle nodes - chosen for second hop in circuit, least restricted set
- Exit nodes - last hop in circuit
  - Visible to outside destination
  - Support filtering of outgoing traffic
  - Most vulerable position of nodes

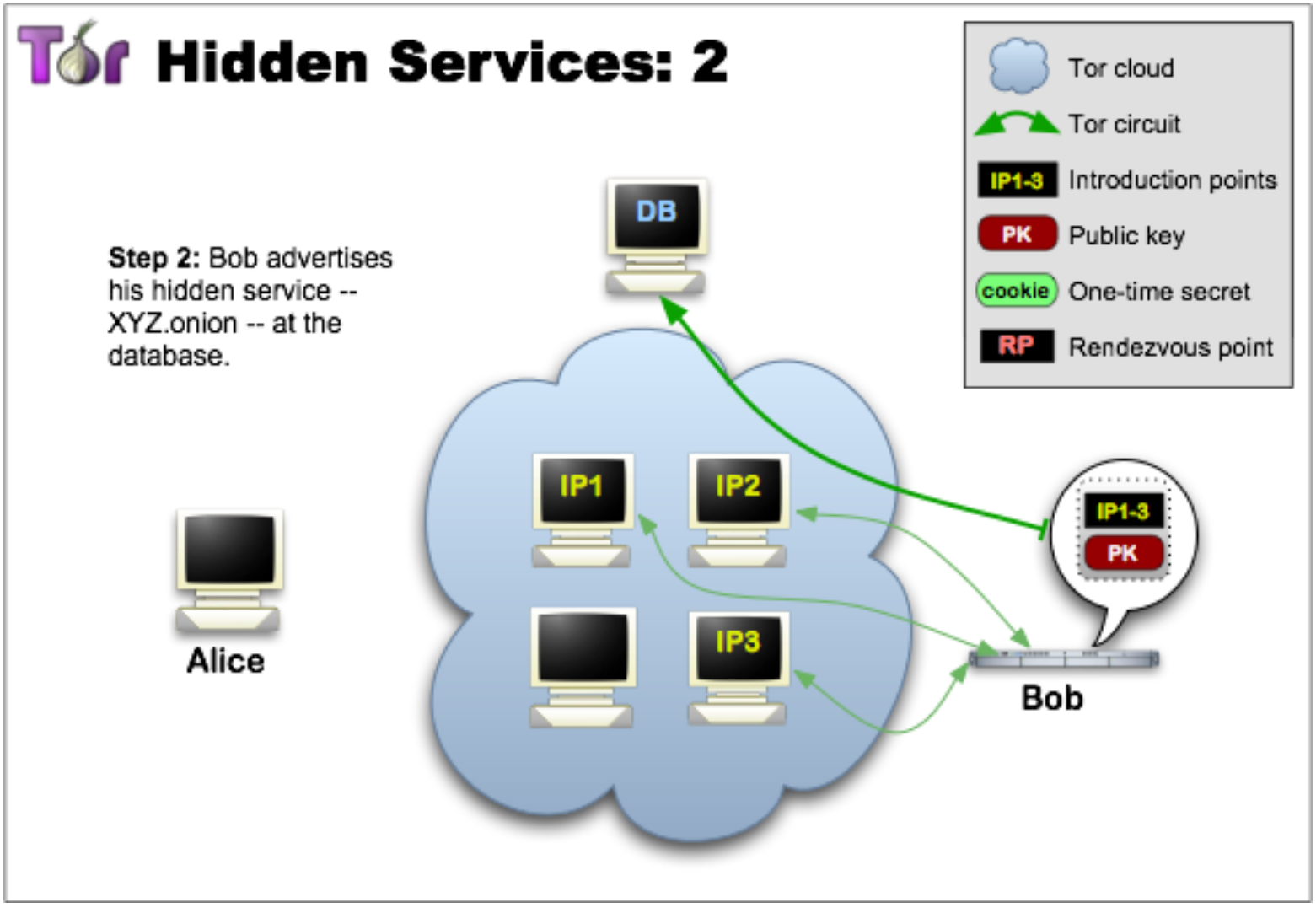
Hidden Services in Tor

- Hidden services allow Tor servers to receive incoming connections anonymously
- Can provide access to services available *only* via Tor
  - Web, IRC, etc.
  - For example, host a website without your ISP knowing

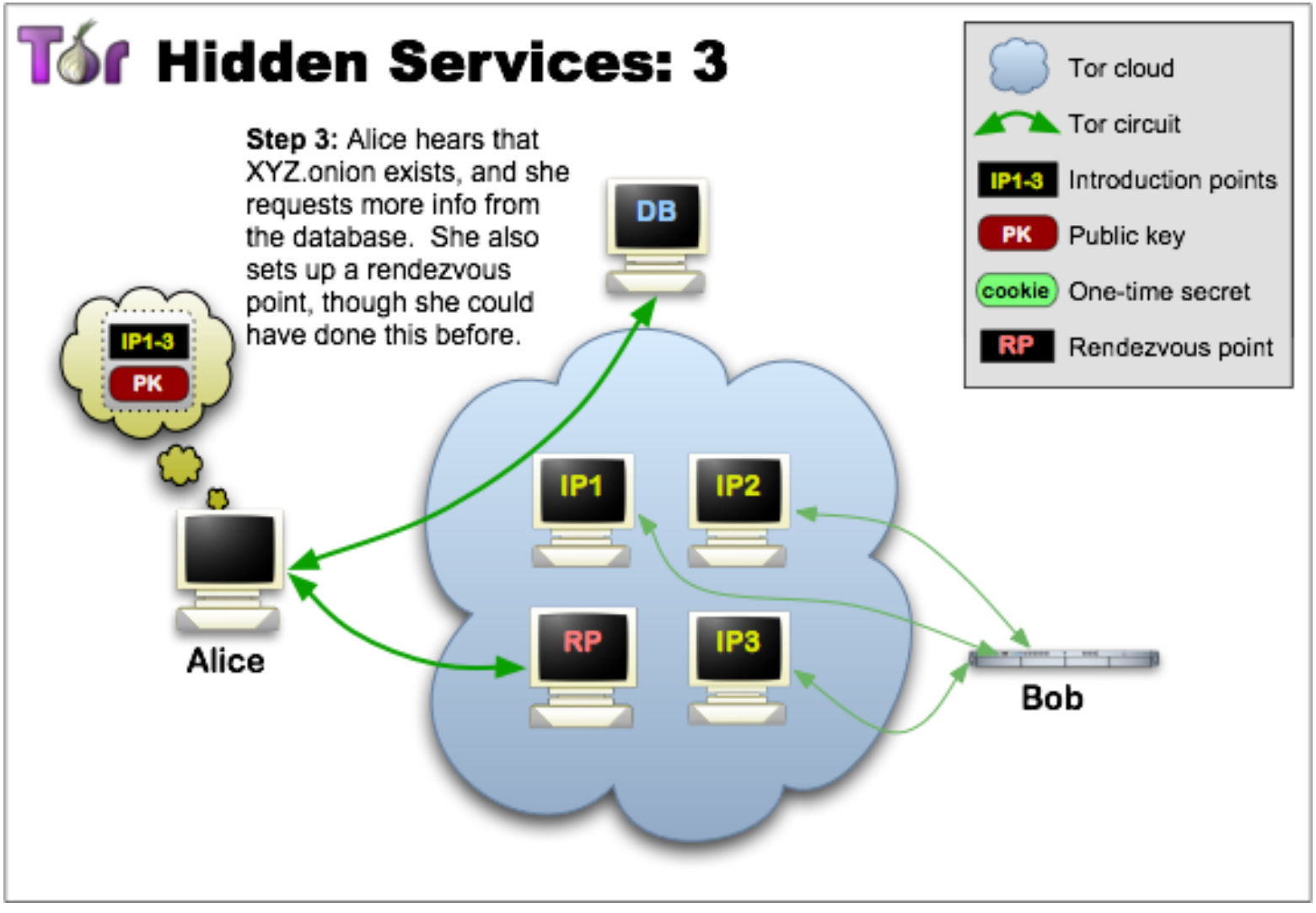
Hidden Services Example 1



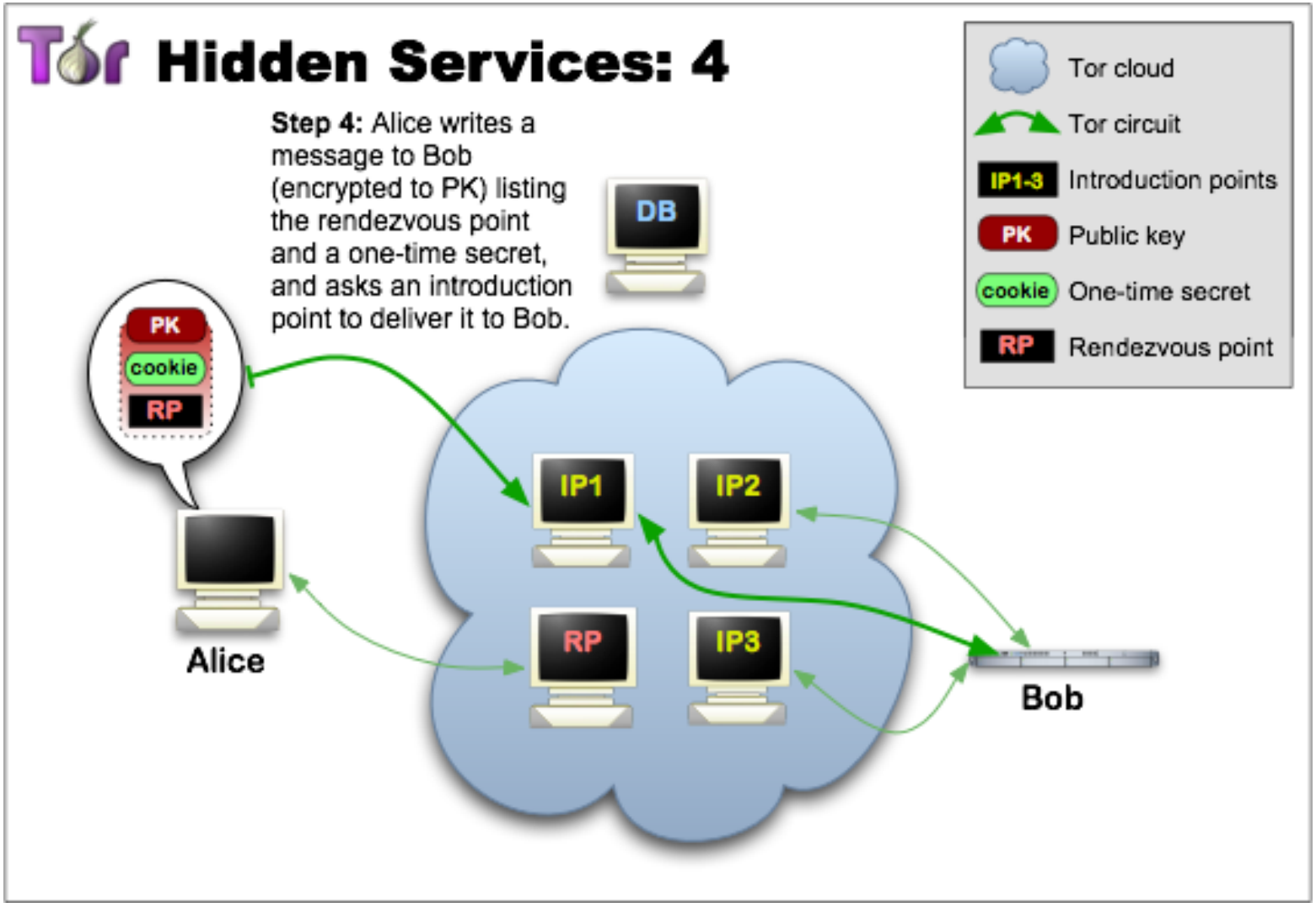
Hidden Services Example 2



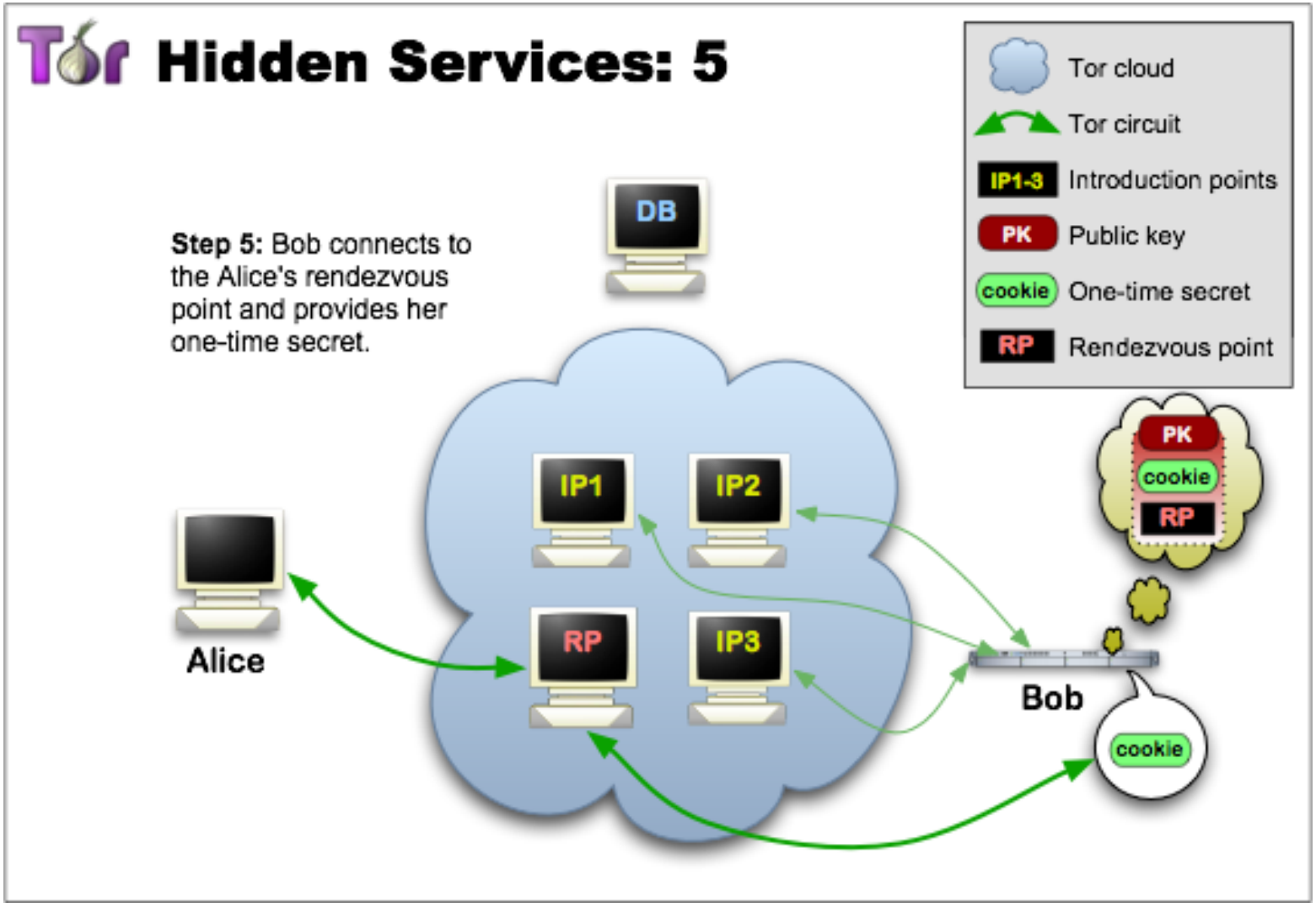
Hidden Services Example 3



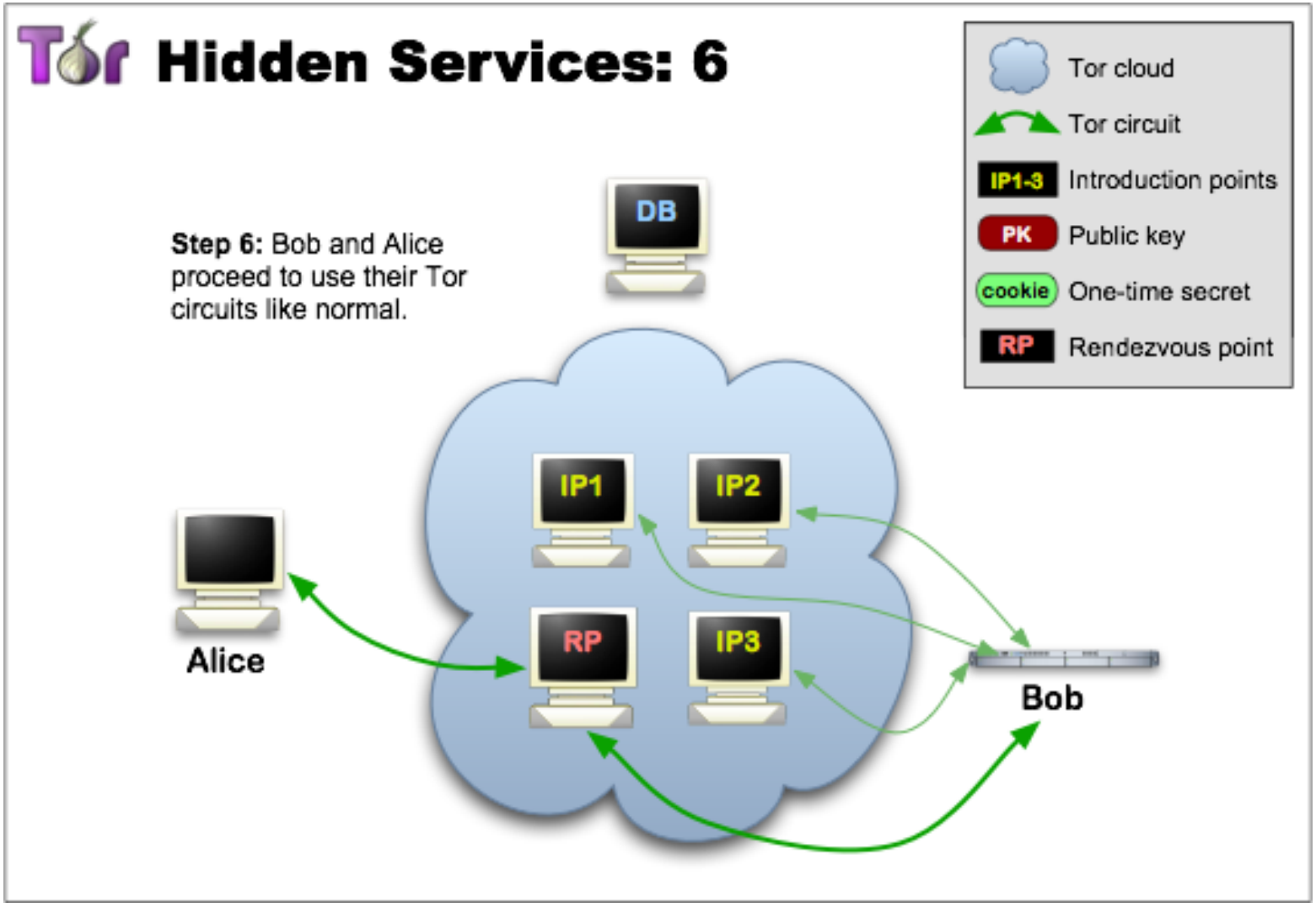
Hidden Services Example 4



Hidden Services Example 5



Hidden Services Example 6



Types of Attacks on Tor

- Exit Relay Snooping
- Website fingerprinting
- Traffic Analysis
- Intersection Attack
- DoS

The Number of Hops

What is more secure: more hops or fewer hops?

Path lifetime

What is more secure:
short-lived paths or long-lived paths?

Pond[3]

- Pseudonymous asynchronous messaging
- Pond servers store (encrypted) messages for multiple users
- User contacts his server to receive, foreign servers to transmit
- Pond communicates only over Tor (Pond server is a hidden service)
- Users make connections periodically, always transferring 16 KB in both directions

[3] `https://pond.imperialviolet.org/`

Pond Forward-Secrecy

- Received messages are deleted after a week by default
- Uses OTR-like encryption with DH rachet (Axolotl)
- Messages always include “next” DH public value
- Users are encouraged to acknowledge receipt

$\Rightarrow$ new DH values!

Pond Spam Protection

- You can *only* receive messages from users you did a key exchange with
- Pond server thus authenticates sender before accepting messages

Pond Spam Protection

- You can *only* receive messages from users you did a key exchange with
- Pond server thus authenticates sender before accepting messages
- Pond uses the BBS group signature scheme for this

$\Rightarrow$ server can check group membership, but not determine sender identity

Pond Storage

- All state in a single encrypted file
- Key derived from user's passphrase, overwritten for every update
- Uses Scrypt to hash passphrase
- Uses TPM NVRAM for erasure storage of key (if available)

Mute/Ants[5]

Properties that a search-limiting mechanism should have:[4]

1. Single Integer Representation
2. Distributed Enforcement
3. Total Limit
4. Deterministic
5. Account for Branching
6. Account for Results

[4] according to Mute author Jason Rohrer

[5] `http://mute-net.sourceforge.net/utilityCounters.shtml`

Utility Counters

UC starts at zero. Without hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| + \beta * |forwardSet| + \gamma$$

Improved formula with hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| * HC + \beta * |forwardSet|^{1+\frac{1}{HC}} + \gamma$$

Utility Counters

UC starts at zero. Without hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| + \beta * |forwardSet| + \gamma$$

Improved formula with hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| * HC + \beta * |forwardSet|^{1+\frac{1}{HC}} + \gamma$$

What is the impact of using UCs for search on anonymity?

Mute Sender Anonymity

Use a hybrid approach for flooding:

- Initiator picks random 20-byte SHA1 hash value
- Each hop re-hashes the current value
- If last bytes is ≤ 51, switch to utility counters

Does this solve the problem?

Mute Responder Anonymity

Use a third approach for the end:

- Forward with branching until UC hits the limit
- Then switch to chain mode
- Each node on startup once determines an operational mode n with probability $p(n)$, and in chain mode forwards to the same n neighbours, where:

$$p(n) = \begin{cases} \frac{3}{4} & n = 0 \\ 2^{-n+2} & n > 0 \end{cases} \tag{9}$$

Does this solve the problem?

GAP

K. Bennett and C. Grothoff introduced GAP [1]:

- uses link-encryted, unstructured overlay network
- search integrated: initiator and responder anonymity
- Simple query-reply scheme:
  - sender queries using hash code
  - responder replies with encrypted reply (ECRS)

GAP

K. Bennett and C. Grothoff introduced GAP [1]:

- uses link-encryted, unstructured overlay network
- search integrated: initiator and responder anonymity
- Simple query-reply scheme:
  - sender queries using hash code
  - responder replies with encrypted reply (ECRS)
- nodes can individually trade anonymity for efficiency
- nodes can not gain anonymity at the expense of other nodes

$\Rightarrow$ "correct" economic incentives

- implemented in GNUnet (fs)

GAP: Money Laundering

If you wanted to hide your financial traces, would you:

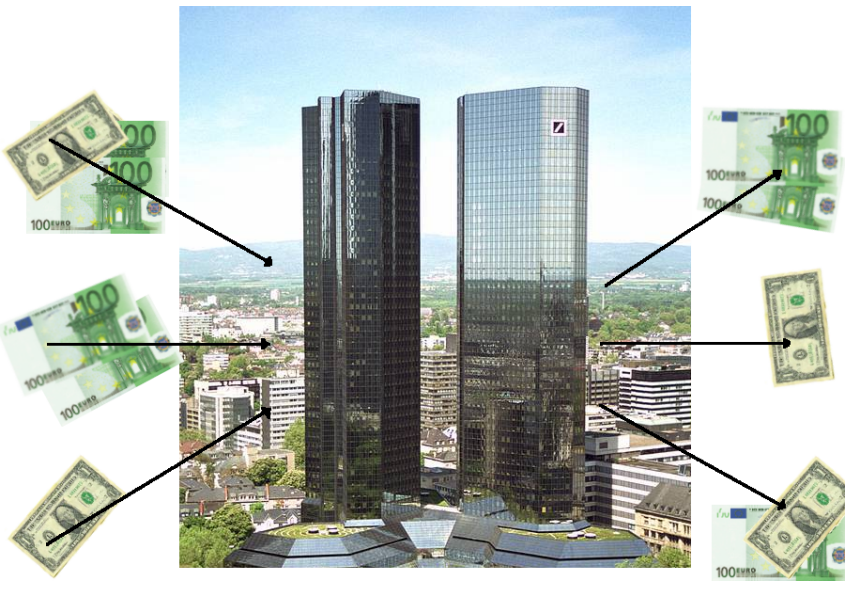
- Give the money to your neighbor,
- expect that your neighbor gives it to me,
- and then hope that I give it to the intended recipient?
- trust everybody involved:
  - to do this for you as a favour,

GAP: Money Laundering

If you wanted to hide your financial traces, would you:

- Give the money to your neighbor,
- expect that your neighbor gives it to me,
- and then hope that I give it to the intended recipient?
- trust everybody involved:
  - to do this for you as a favour,
  - not to steal the money, and
  - not to tell the police?

GAP: Banks!



GAP: key idea

Source rewriting was traditionally used to hide the identity of the source. GAP uses it in a different way:

- Anonymity is achieved by making the initiator look like a router that acts on behalf of somebody else
- It is important to make traffic originating from the router look identical to traffic that the router indirects
- It is **not** necessary to avoid a direct network connection between the responder and the initiator

GAP: Why indirect?

- Indirections do not protect the sender or receiver
- Indirections can help the indirector to hide its own traffic
- If the indirector cheats (e.g. by keeping the sender address when forwarding) it only exposes its own action and does not change the anonymity of the original participants

GAP: Key Realization

Anonymity can be measured in terms of

- how much traffic from non-malicious hosts is indirected compared to the self-generated traffic
- in a time-interval small enough such that timing analysis can not disambiguate the sources.

GAP routing: Local Heuristics

In GAP, each peer is free to route queries whichever way it thinks is best.

- structured routing is **predictable** and **analyzable**
- GAP keeps routing hard to predict, peers do not disclose information

GAP routing: Local Heuristics

In GAP, each peer is free to route queries whichever way it thinks is best.

- structured routing is **predictable** and **analyzable**
- GAP keeps routing hard to predict, peers do not disclose information
- **proximity**-based routing is **efficient** for **migrated** content
- **hot-path** routing is **efficient** if queries are **correlated**
- **flodding** is acceptable if merely **noise** is **substituted**

GAP routing: Local Heuristics

In GAP, each peer is free to route queries whichever way it thinks is best.

- structured routing is **predictable** and **analyzable**
- GAP keeps routing hard to predict, peers do not disclose information
- **proximity**-based routing is **efficient** for **migrated** content
- **hot-path** routing is **efficient** if queries are **correlated**
- **flodding** is acceptable if merely **noise** is **substituted**

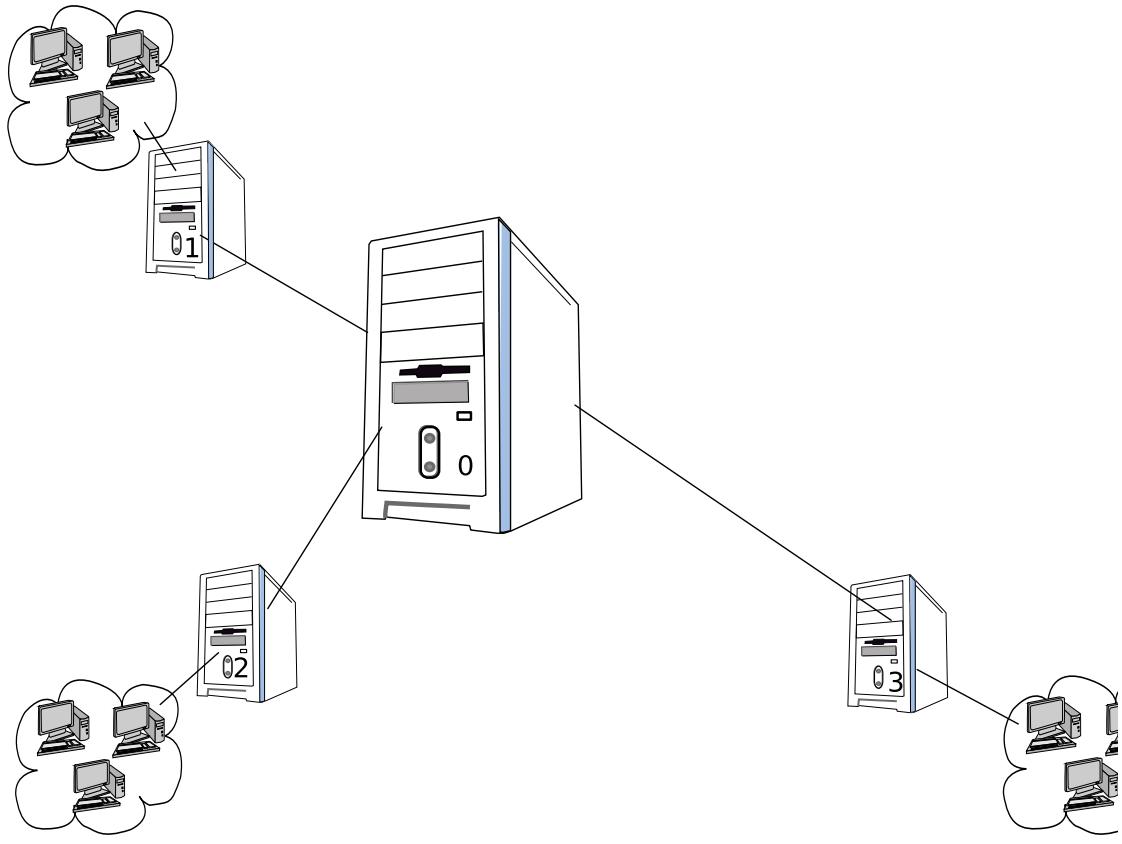
How long should a peer keep track of which queries?

Time-to-Live

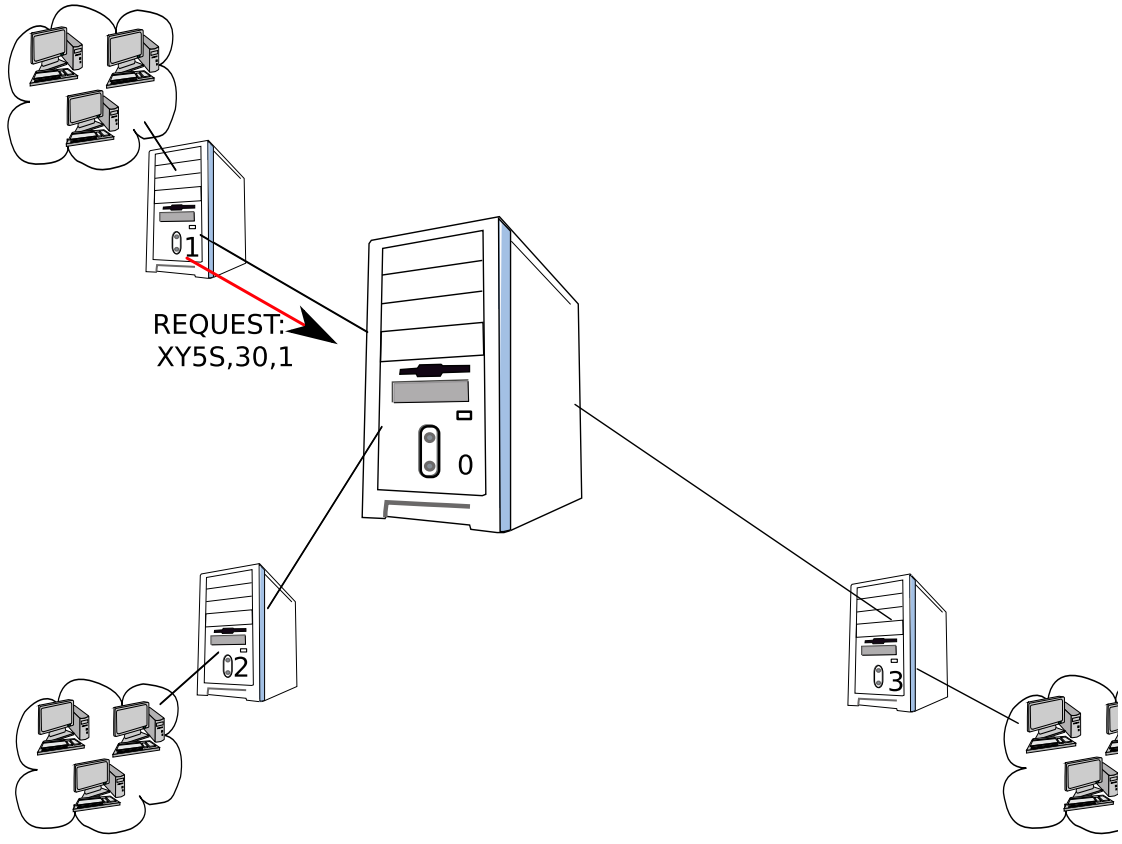
- TTL field in queries is **relative time** and can be **negative**.
- Absolute TTL = NOW + relative TTL
- Absolute TTL and decies which query to **drop**.
- TTL is decremented at each hop.
- peers can still route "expired" queries indefinitely

$\Rightarrow$ better solution than traditional hop-count

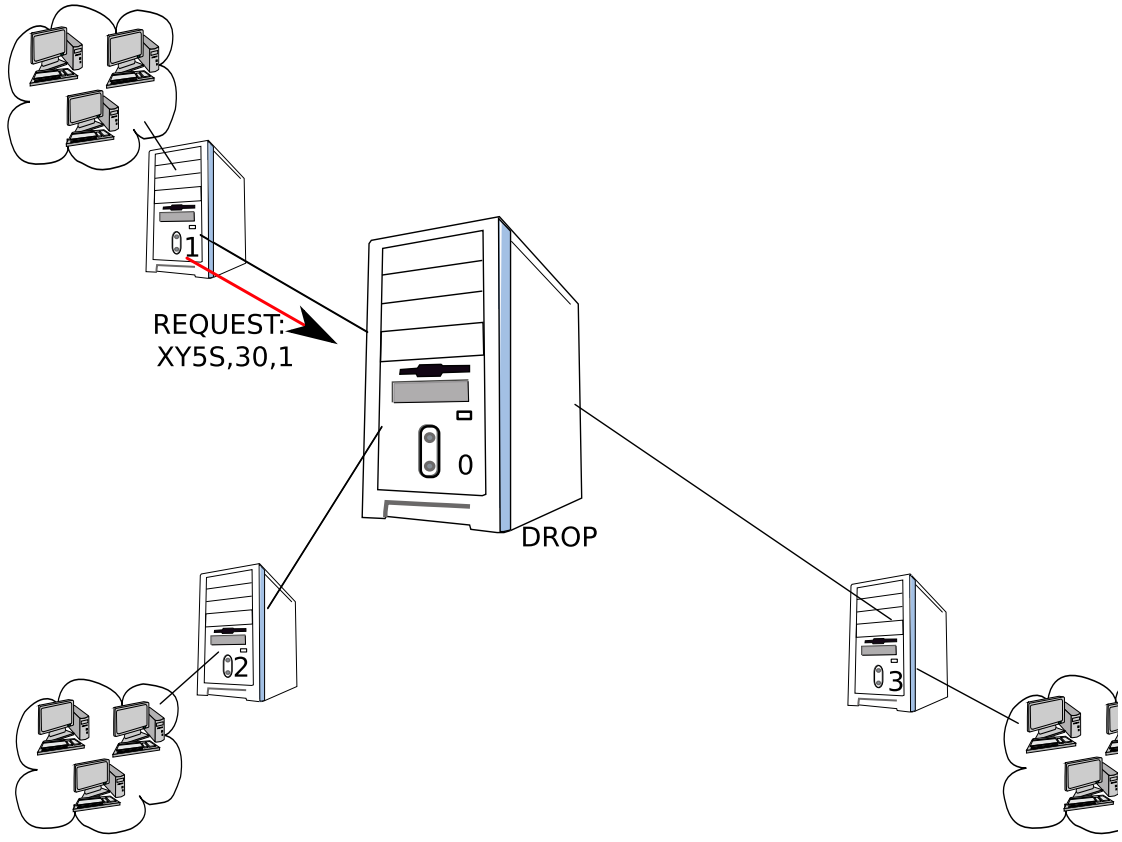
GAP illustrated (1/9)



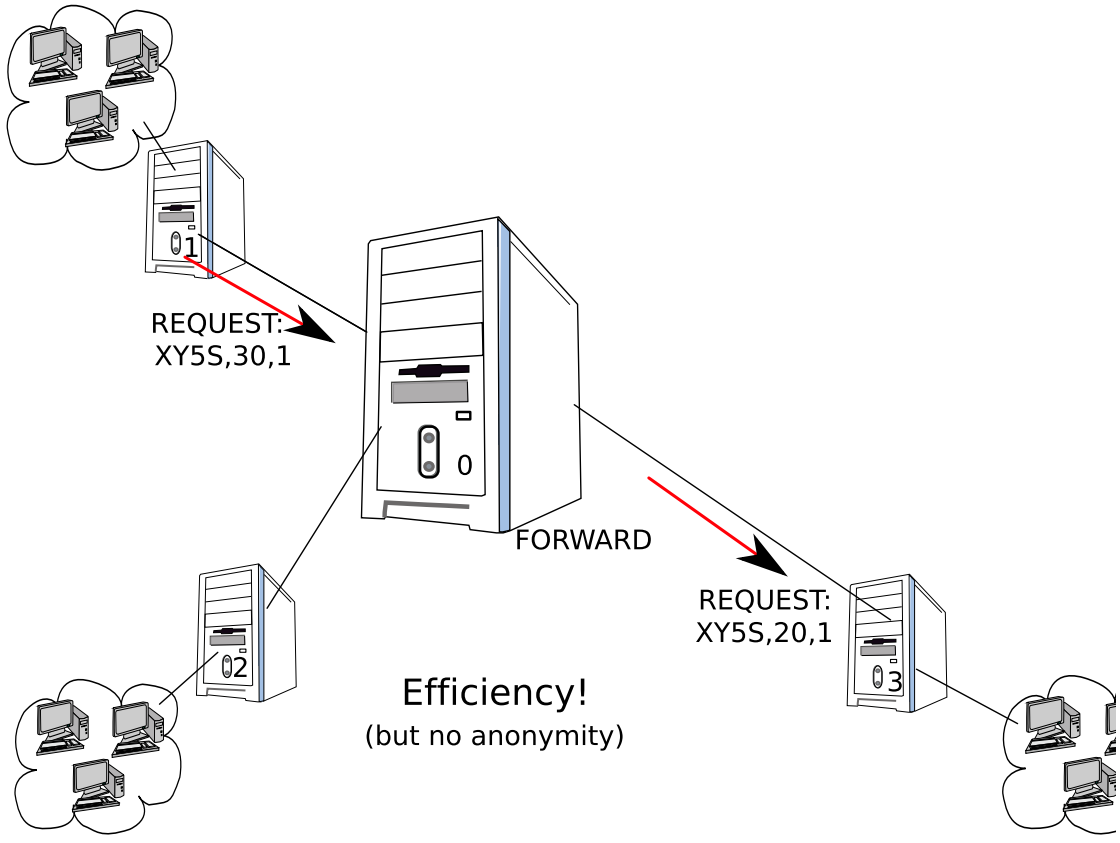
GAP illustrated (2/9)



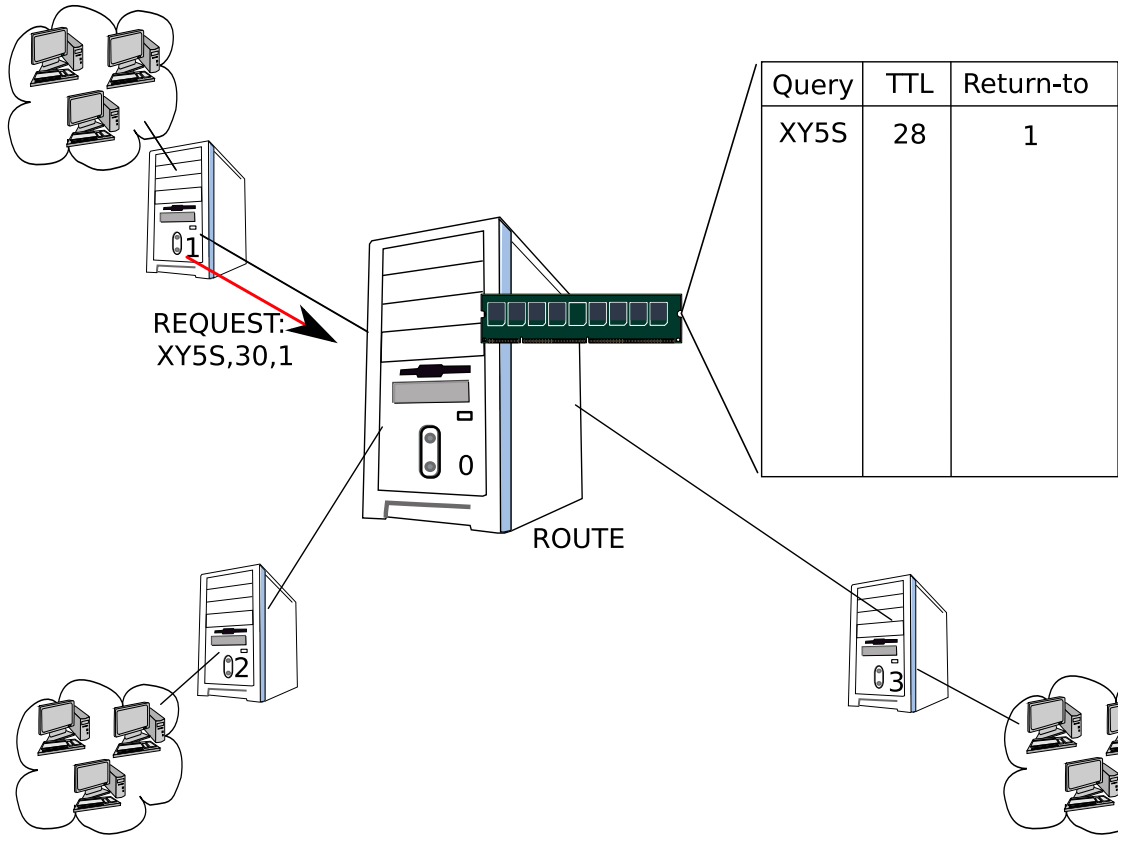
GAP illustrated (3/9)



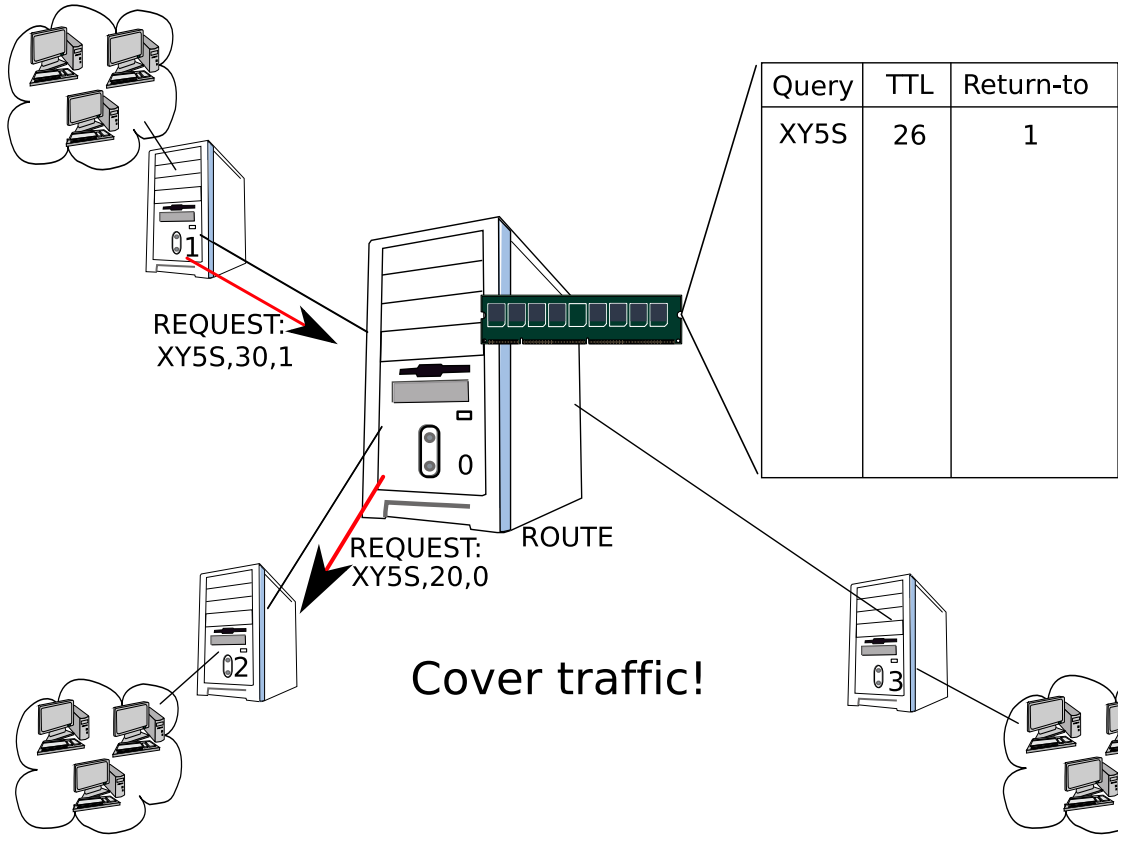
GAP illustrated (4/9)



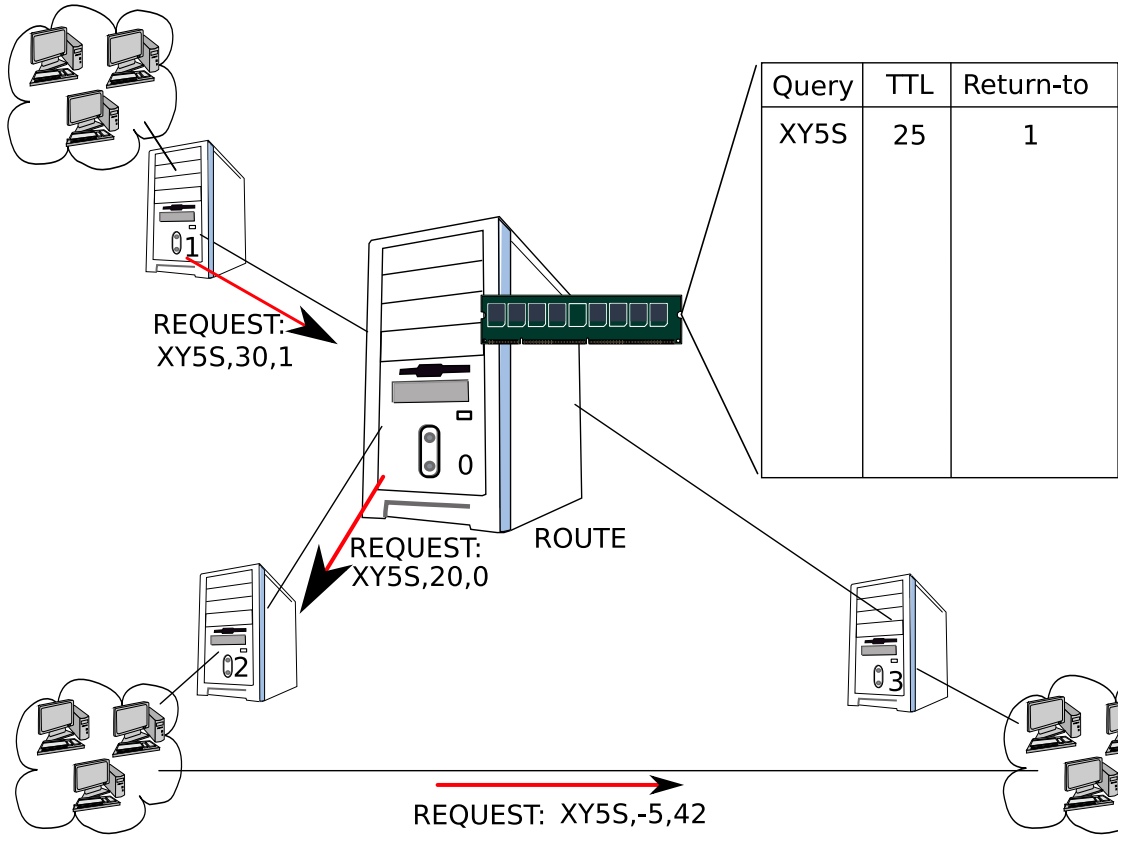
GAP illustrated (5/9)



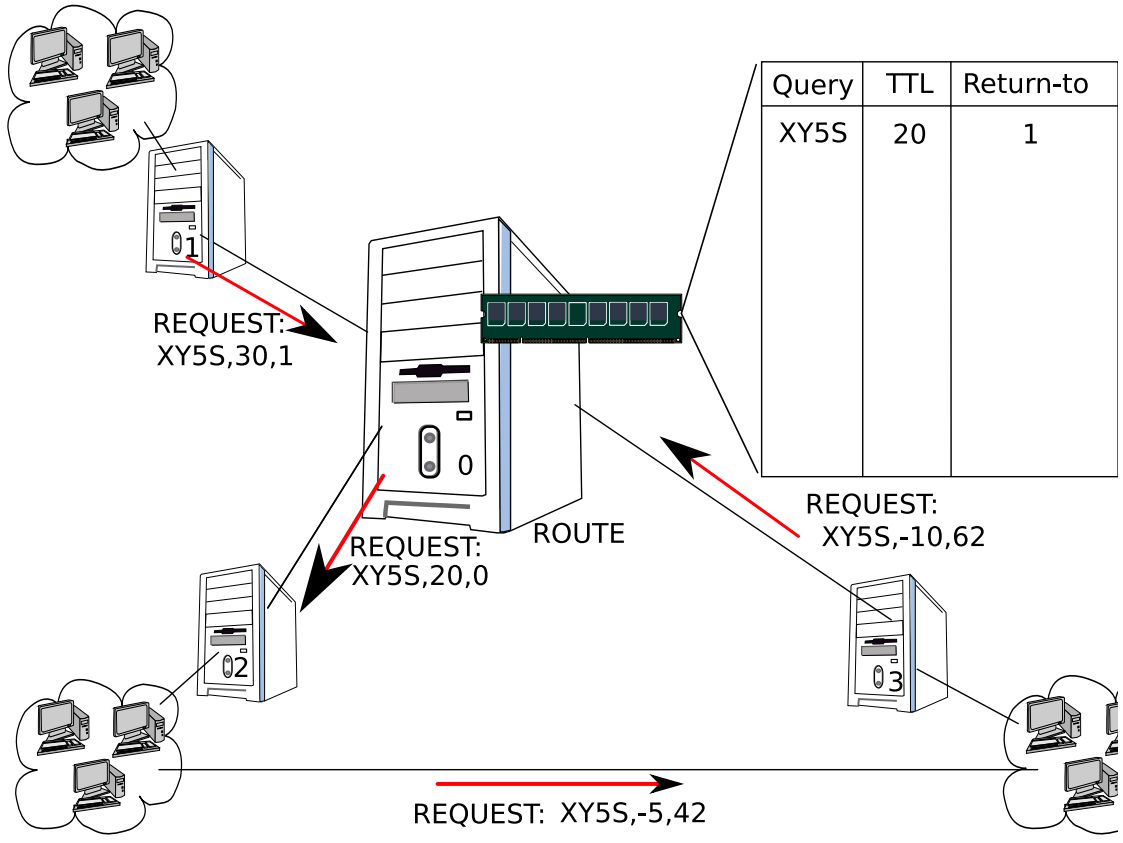
GAP illustrated (6/9)



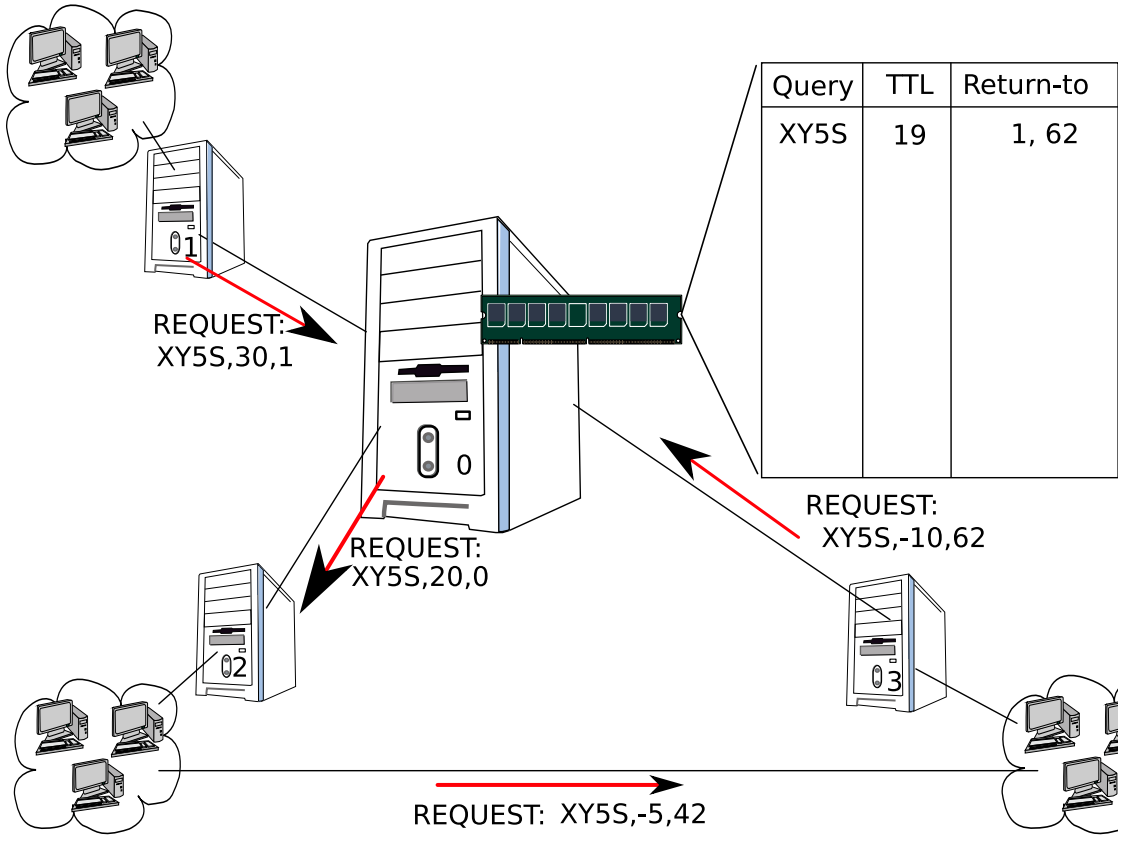
GAP illustrated (7/9)



GAP illustrated (8/9)



GAP illustrated (9/9)



GAP: efficient or anonymous

When a node M processes a query from A, it can choose:

- to how many other nodes C_i should receive the query
- to tell C_i to send the reply directly to A
- to send a reply if content is available

GAP: efficient or anonymous

When a node M processes a query from A, it can choose:

- to how many other nodes C_i should receive the query
- to tell C_i to send the reply directly to A
- to send a reply if content is available

If a node forwards a query preserving the identity of the originator, it may *expose* the actual initiator to the responder. This is ok:

- Next hop has still no certainty that the exposed predecessor is not routing for somebody else
- Same argument holds for the other direction

Costs and benefits of short-cuts

By preserving the previous sender of the query when the short-cutting peer forwarded the query:

- the peer has exposed its own routing behaviour for this message, reducing the set of messages it can use to hide its own traffic
- the peer has gained performance (bandwidth) since it does not have to route the reply

Costs and benefits of short-cuts

By preserving the previous sender of the query when the short-cutting peer forwarded the query:

- the peer has exposed its own routing behaviour for this message, reducing the set of messages it can use to hide its own traffic
- the peer has gained performance (bandwidth) since it does not have to route the reply

A node decides to forward a query based on the current load:

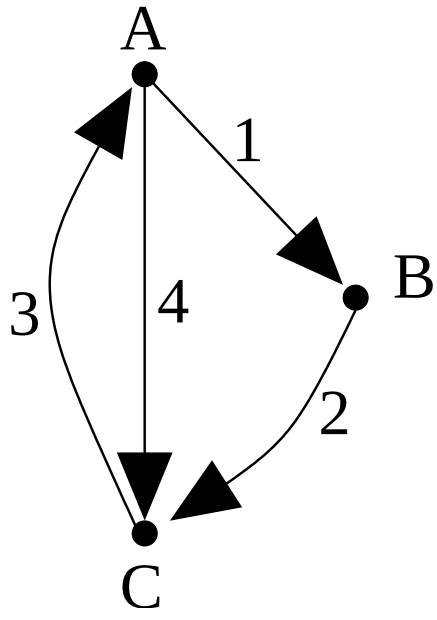
- if the load is low, the node maximizes the indirected traffic and thus its anonymity
- if the load is high, the node is already covered in terms of anonymity and it reduces its load (does not have to route the replies) by forwarding
- if the load is far too high, the node just drops packets.

GAP: individual trade-offs

In summary:

- indirect when idle,
- forward when busy,
- drop when very busy.

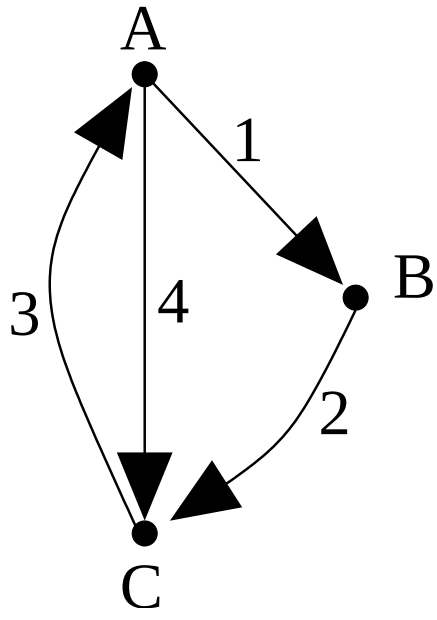
If we are handling too much traffic, we likely do not need as much to hide ourselves and can be *more efficient*!



GAP: individual trade-offs

In summary:

- indirect when idle,
- forward when busy,
- drop when very busy.

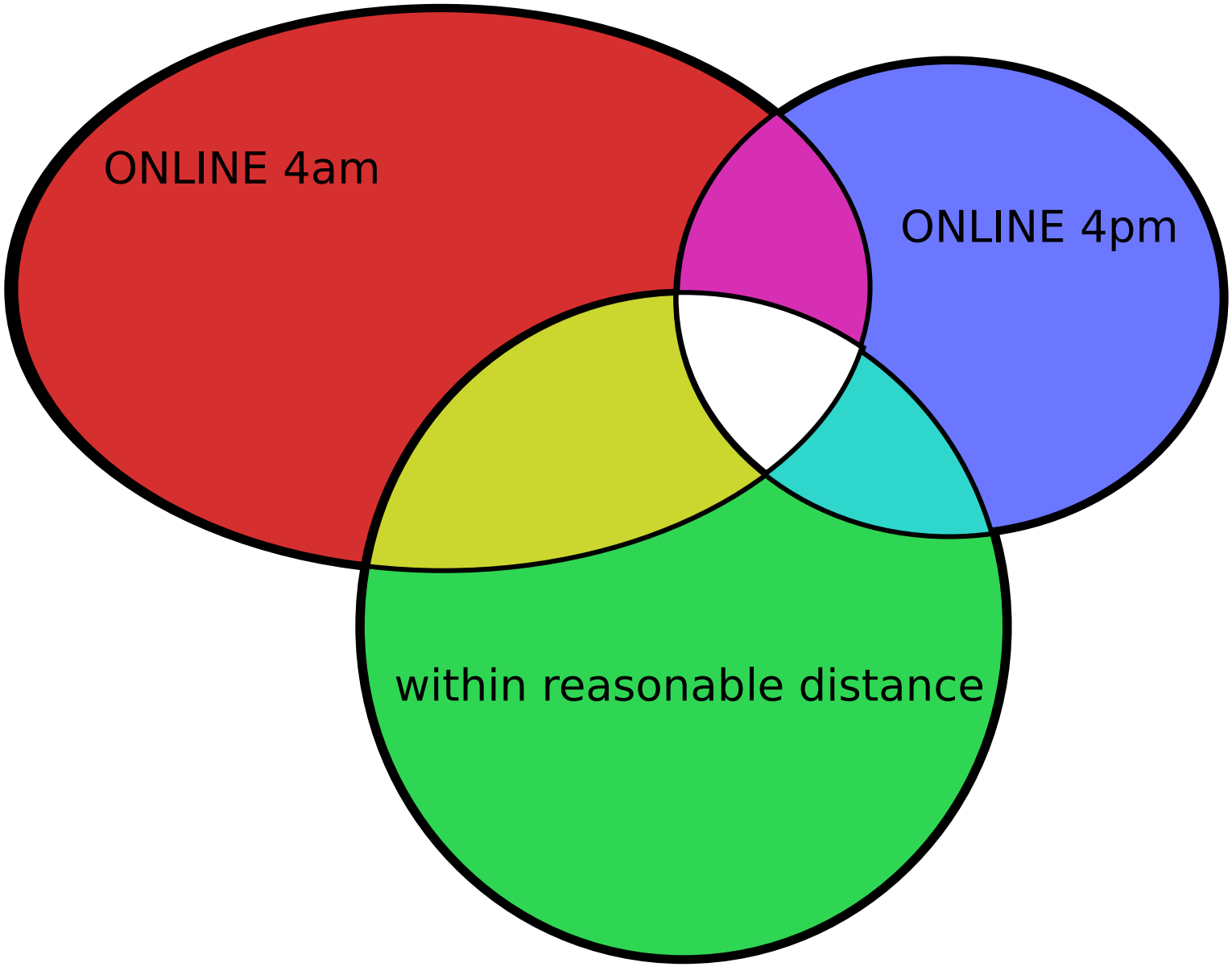


If we are handling too much traffic, we likely do not need as much to hide ourselves and can be *more efficient*!

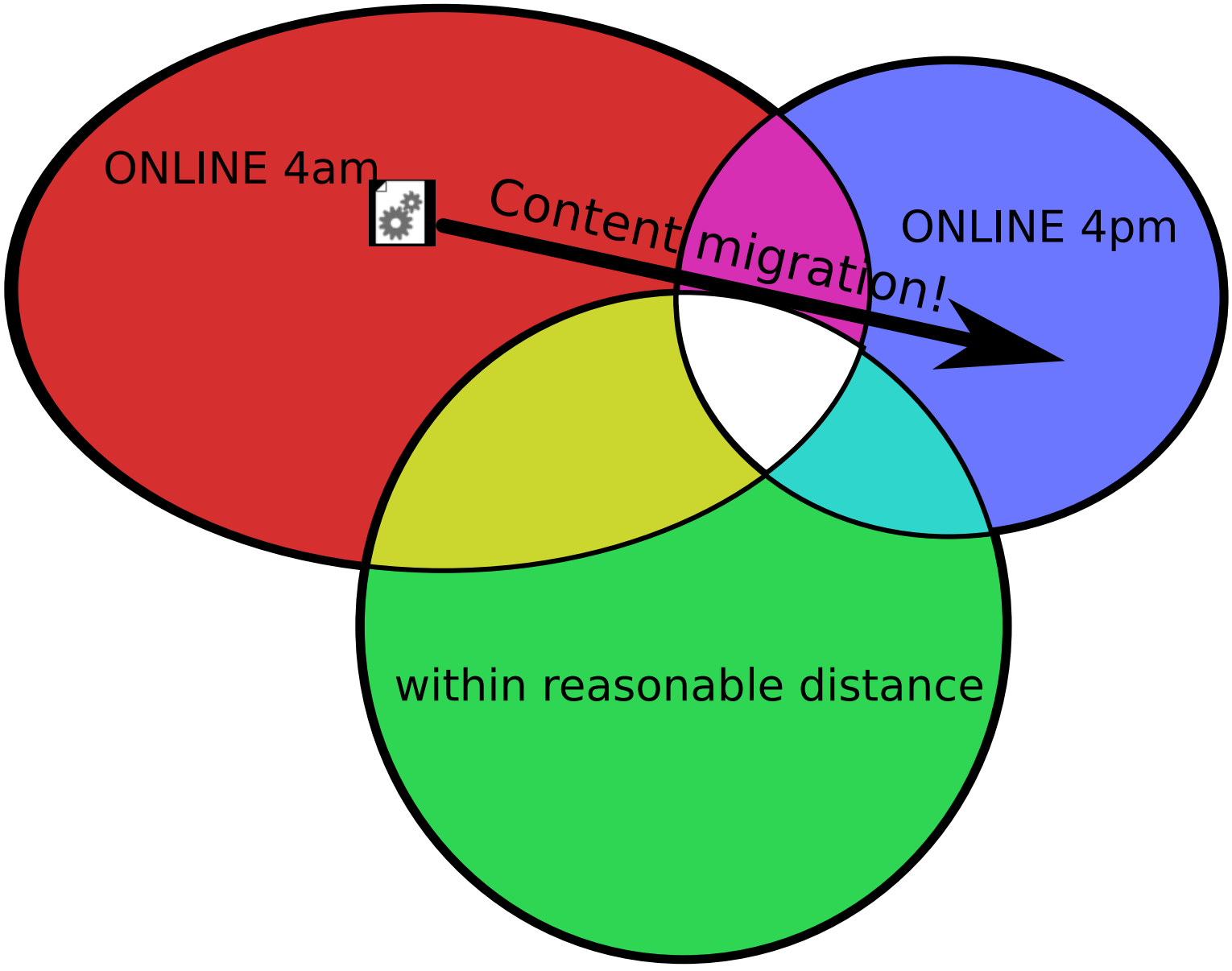
GAP is unreliable and has best-effort semantics:

- packets can be lost, duplicated or arrive out-of-order
- nodes can act more randomly and adjust to load
- application layer is responsible for adding reliability

Attacks: Partitioning (1/2)



Attacks: Partitioning (2/2)



GAP: Traffic Analysis?

A powerful adversary doing traffic analysis sees:

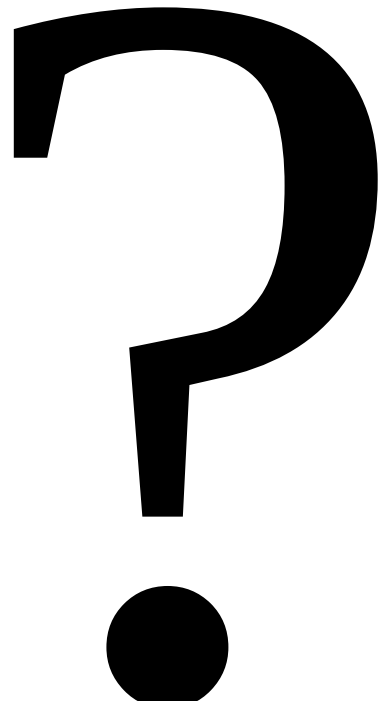
- encrypted packets
- unlinkable queries or replies at collaborating nodes
- random delays, unpredictable packet drops
- unpredictable packet duplication
- only a small part of the network's topology since no routing information is exchanged

GAP: Conclusion

GAP can achieve:

- any degree of anonymity based on the bandwidth available to the user compared to the adversary
- scalability because busy nodes can increase thoughput without compromising anonymity

Questions?



"A society that gets rid of all its troublemakers goes downhill."
–Robert A. Heinlein

References


Krista Bennett and Christian Grothoff.
gap - Practical Anonymous Networking.
In *Designing Privacy Enhancing Technologies*, pages 141–160. Springer-Verlag, 2003.


George Danezis, Roger Dingledine, and Nick Mathewson.
Mixminion: Design of a type iii anonymous remailer protocol.
In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, 2003.


Brad Miller, Ling Huang, A.D. Joseph, and J.D. Tygar.
I know why you went to the clinic: Risks and realization of https traffic analysis.
`http://arxiv.org/abs/1403.0297`, 2014.


Michael K. Reiter and Aviel D. Rubin.
Anonymous web transactions with crowds.
Commun. ACM, 42(2):32–48, February 1999.