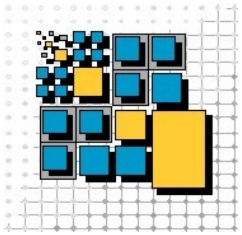
Anonymity in P2P Systems

Protecting User Presence by Hiding Tor Hidden Service Activity

Karsten Loesing

Distributed and Mobile Systems Group, University of Bamberg

1. Bamberger-Zwickauer Workshop, 2007-06-14



What is user presence?

Protecting User Presence by Hiding Tor Hidden Service Activity

- Instant Messaging systems allow exchange of user presence and text messages
- User presence is the knowledge whether a communication partner is likely to answer before contacting him
- Focus on boolean user presence information vs. additional awareness information
- Problem not limited to IM, further apps featuring presence-awareness imaginable

Why is user presence something protectable?

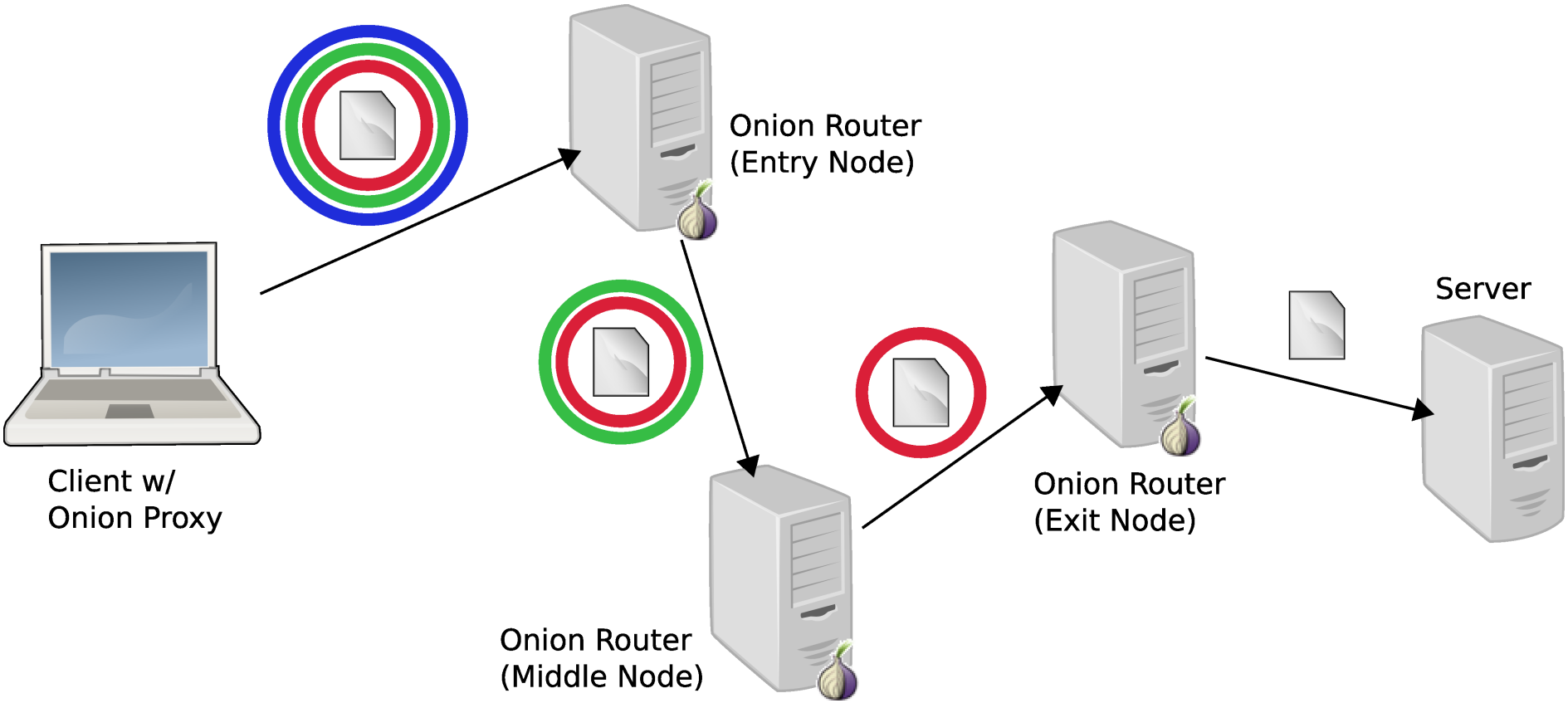
Protecting User Presence by Hiding Tor Hidden Service Activity

- Track someone's online activity
- Guess time-zone
- Derive patterns
- Observe deviations
- Conclude personal behavior
- ...
- Idea: No trust in system provider, pass user presence only to buddies!

Tor (The Onion Router)

Protecting User Presence by Hiding Tor Hidden Service Activity

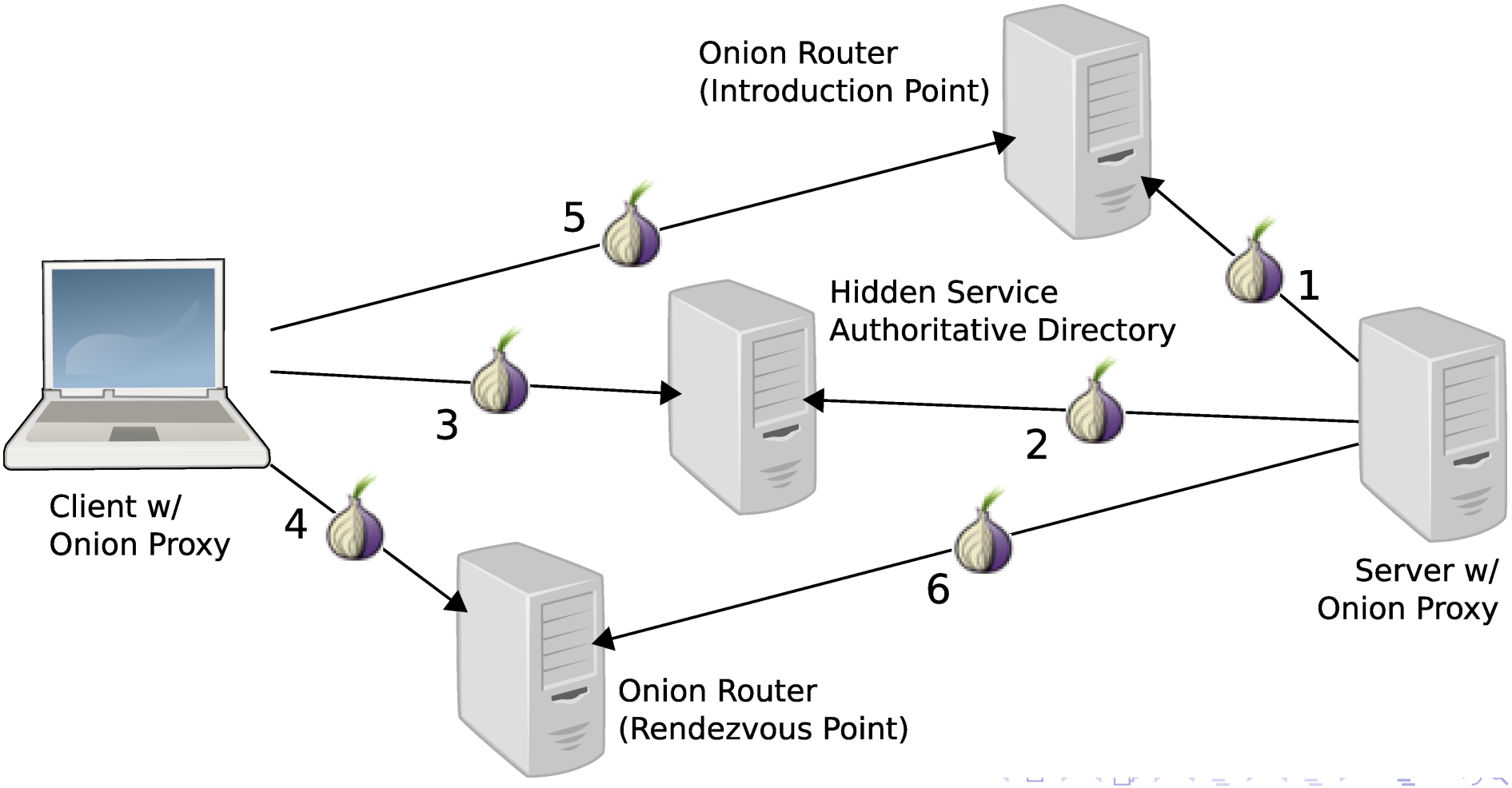
- Tor hides IP addresses



Tor Hidden Services

Protecting User Presence by Hiding Tor Hidden Service Activity

- Tor hidden services make it possible to advertise a service without telling its IP address



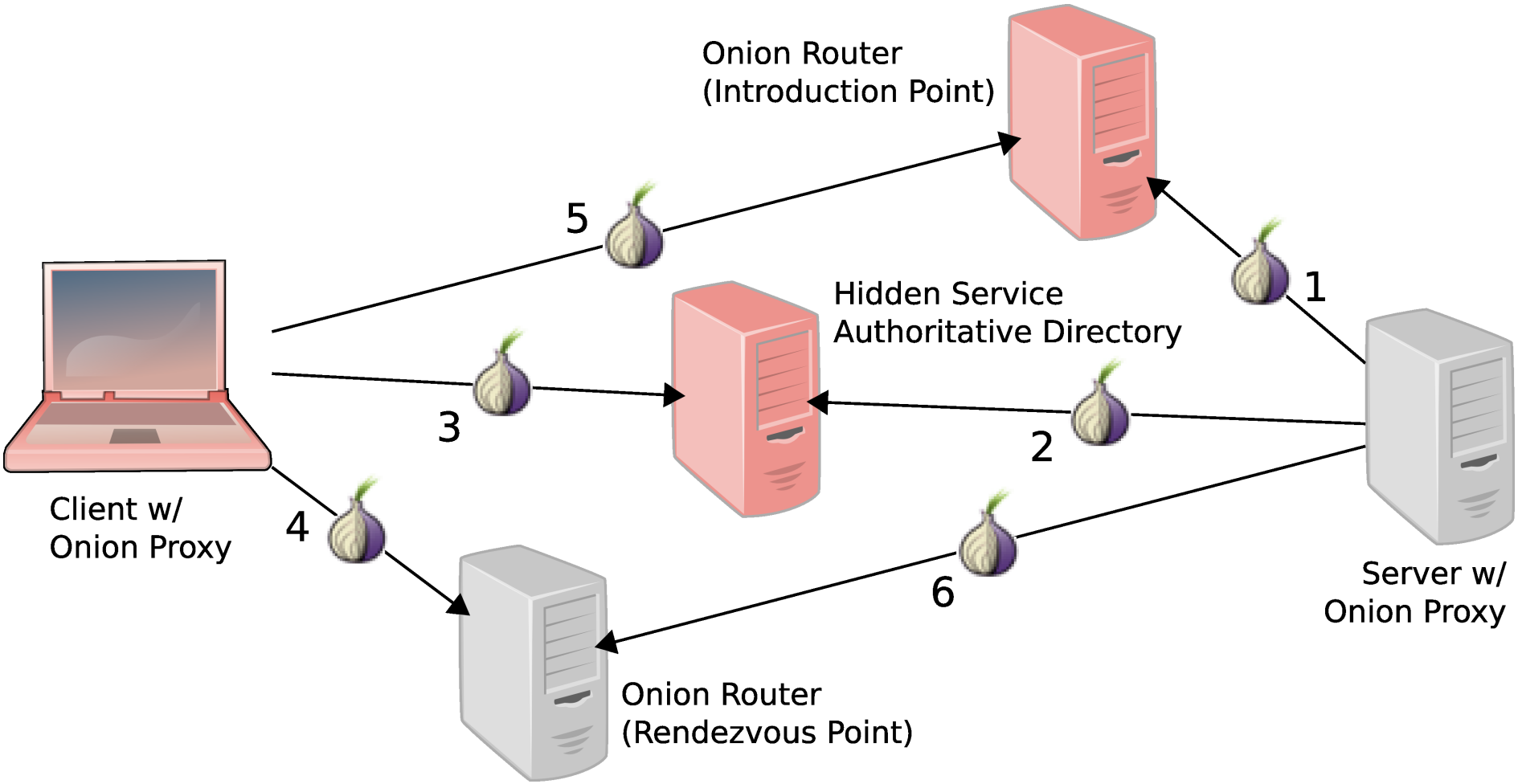
How can Tor hidden services protect user presence?

- IM user configures Tor hidden service and advertises onion address to buddies
- Buddies establish connection via Tor
- User and buddy exchange presence information and text messages

Problem: Activity reveals presence

Protecting User Presence by Hiding Tor Hidden Service Activity

- Tor does not yet intend to hide the activity of a hidden service



What is necessary to hide activity?

1. Modify descriptor format
2. Distribute storage of descriptors
3. Change protocol to establish introduction points

Step 1: Modify descriptor format

Definition

`descriptor-id = h(public-key)`

- Rendezvous Service Descriptor (descriptor) contains contact information for clients (list of introduction points)
- `descriptor-id` used for storage and lookup of current descriptor in Tor directory
- `descriptor-id` derived from `public-key` of hidden service: provides authenticity
- Problem: Publication of descriptor with `descriptor-id` reveals activity of hidden service (and fetching reveals usage)

New lookup key

Definition

```
descriptor-id = h(h(public-key) + h(date + cookie))
onion-address = h(public-key) + cookie
```

- Separation of `descriptor-id` (used for lookup) and `onion-address` (told to clients)
- `descriptor-id` of a hidden service (`public-key`) needs to change frequently (`date`) and unpredictably for non-clients (`cookie`)
- Current `descriptor-id` can be constructed by both, server and clients without interaction (only symmetric cryptography)
- Include `h(date + cookie)` to `descriptor-content` so as to verify authenticity of `descriptor-id` without `cookie`

Problem with descriptor content

Definition

```
descriptor-content = {
  public-key,
  h(date + cookie),
  timestamp,
  introduction-points
} signed with private-key
```

- `public-key`:
  - Required to verify descriptor content,
  - authorize use of `descriptor-id`, and
  - encrypt initial message to hidden service.
  - But: reveals hidden-service activity to directory nodes
- `h(date + cookie)`: allows verification of `descriptor-id`
- `timestamp`: ensures freshness
- `introduction-points`: provide up-to-date contact information

Can't we simply encrypt the content with `cookie`?

Example

```
descriptor-id = 6sxoyfb3h2nvok2d6sxoyfb3h2nvok2d,
descriptor-content = <encrypted>
```

- Encryption not possible, because storing nodes could not verify the origin (provider) and filter false entries
  - would make DoS with random entries easy
  - would allow DoS performed by (former) client who is able to generate descriptor ID

Encrypt introduction points

Definition

```
descriptor-id = h(h(public-key) + h(date + cookie))
descriptor-content = {
  public-key,
  h(date + cookie),
  timestamp,
  { introduction-points } encrypted with cookie
} signed with private-key
```

- Encrypt `introduction-points`: useful to prevent DoS attacks, enables hidden-service authentication
- Leave the rest unencrypted:
  - `public-key` and `h(date + cookie)` required to verify `descriptor-id`,
  - `public-key` necessary to verify `descriptor-content`, and
  - `timestamp` used to check freshness.
- But: `public-key` still reveals hidden-service activity

Step 2: Distribute storage of descriptors

- Distribute storage among large set of nodes (Tor onion routers)
- Use DHT-like structure based on existing Tor router list (avoids maintenance messages for routing information)
- Replicate descriptors (on non-consecutive nodes; black-hole problem, still open) to resist node failures and dishonest nodes
- Makes revelation of service activity very hard
  - Probability for observing certain descriptor (per day): $p = 1 - \frac{\binom{N-c}{r}}{\binom{N}{r}}$ with N (total number of nodes), c (number of corrupt nodes), r (number of replicated descriptors).
  - Potential to track service activity increases with number of replicas
- Increases service availability
  - Probability to control all replicas of a descriptor (per day): $p = \frac{\binom{c}{r}}{\binom{N}{r}}$
  - Service availability increases with number of replicas

Average number of nodes

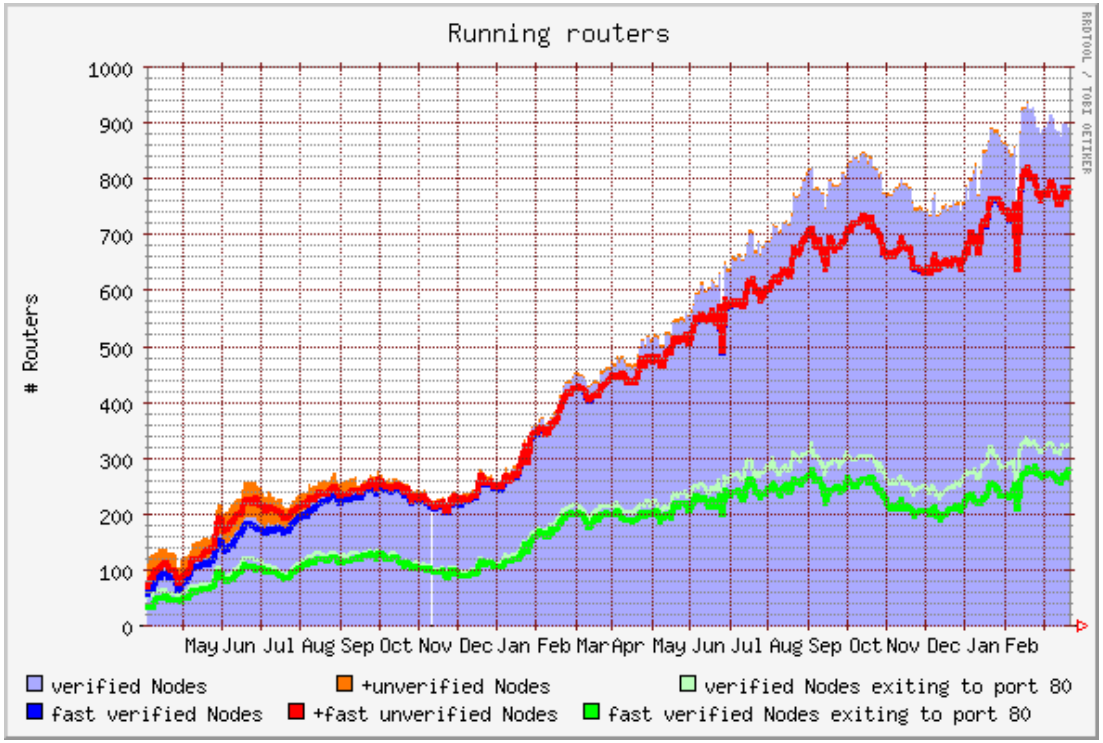


Figure: Graph of the number of Tor servers over the last 24 months. (Source: http://www.noreply.org/tor-running-routers/totalLong.html, March 2007)

Session time

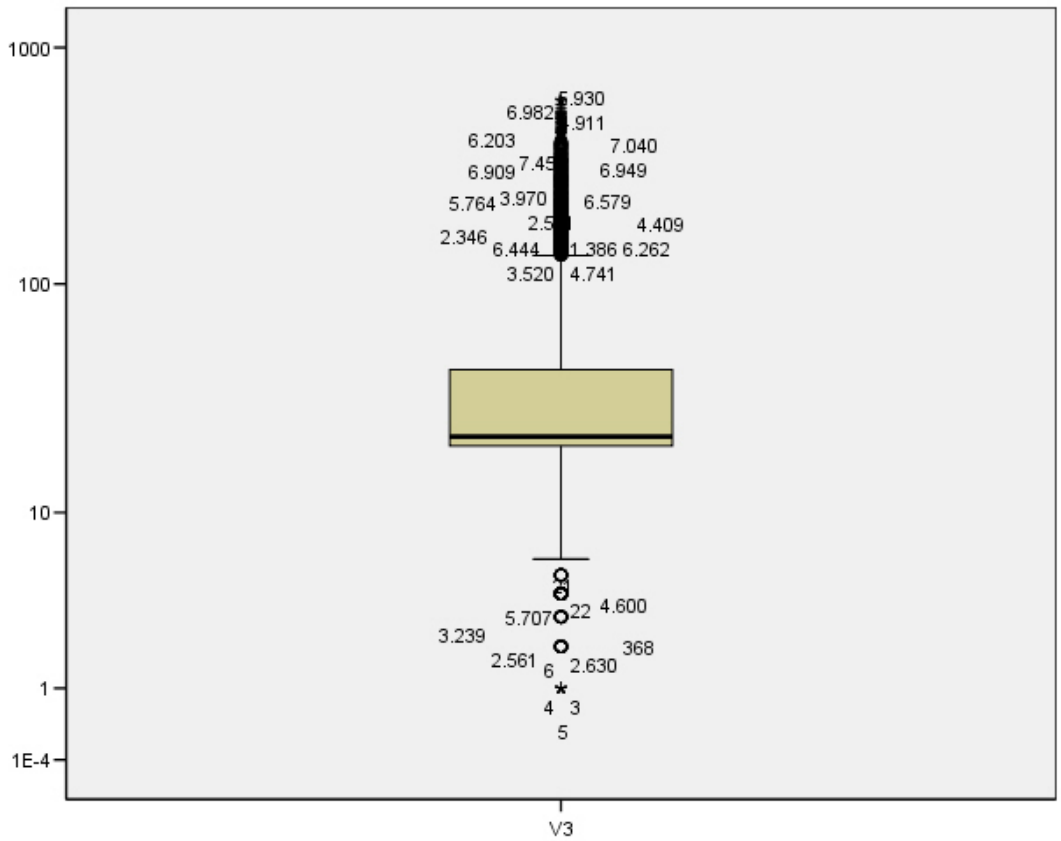


Figure: Box plot of session times in hours with a logarithmic scale. (Evaluation of publicly available log files)

Churn rate

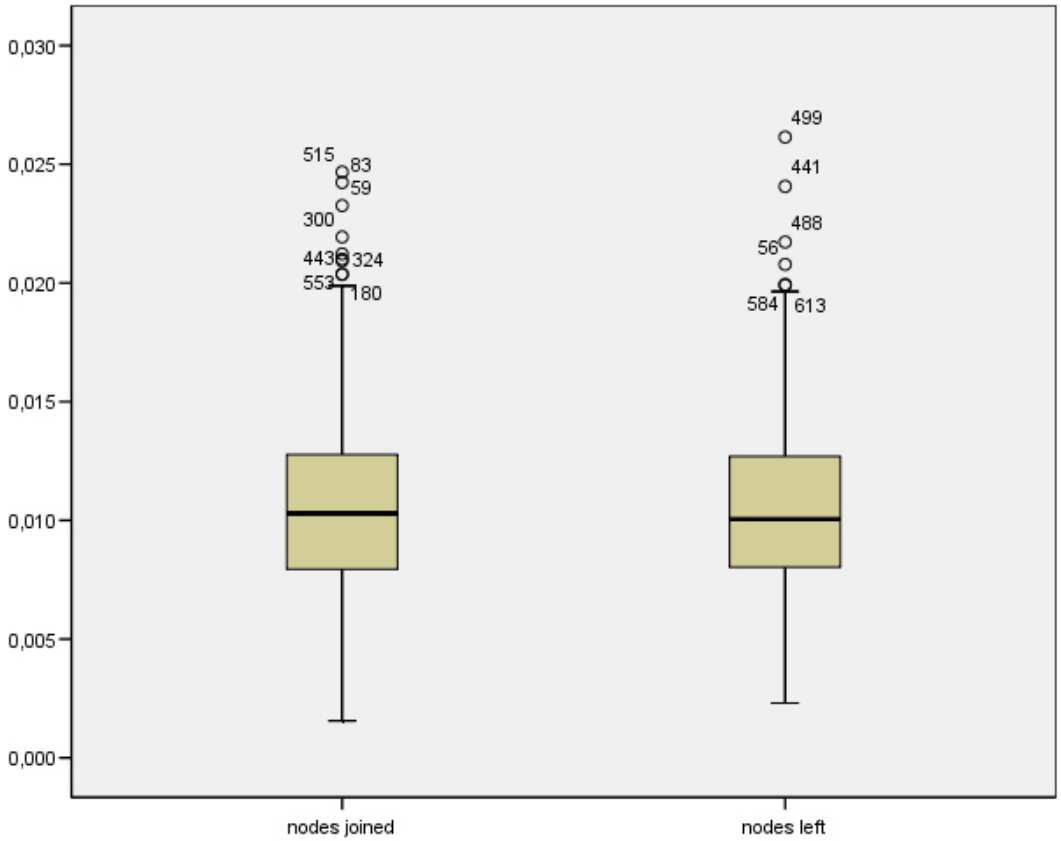


Figure: Box plot of join and leave rates, i.e. the number of joining and leaving nodes per hour compared to the whole node population. (Evaluation of publicly available log files)

Estimated load

- In 15-minute interval:
  - Total number of publish requests: 363.2 ± 65.6
  - ... of which are novel services: 0.8 ± 1.2
  - Total number of fetch requests: 28.9 ± 12.7
  - ... of which can be answered successfully: 15.9 ± 7.4
- Total number of descriptors: $\approx 1,000$
- Numbers expected to increase when (currently poor) performance of hidden services improves
- Statistics collected by (legal) code modification on central Tor directory node

Step 3: Change protocol to establish introduction points

- Last but not least: introduction points don't need to know hidden service activity!
- Hide away service activity from introduction points
- Use fresh service key instead of public key of hidden service
- Include service key in encrypted `introduction-points`

Current status

- Done:
  - Preliminary work (feasibility of nodes for DHT, estimation of load)
  - Tor proposal #114 currently under discussion in public mailing list
  - Java-based test environment to create local Tor network (PuppeTor)
  - Implementation of encoding/parsing new descriptor in C (step 1)
- To be done in the next weeks/months:
  - Implementation of distributing descriptors (step 2, major part of coding)
  - Implementation of changed protocol to establish introduction points (step 3)
- Future work:
  - Write PhD thesis about it...

Questions...