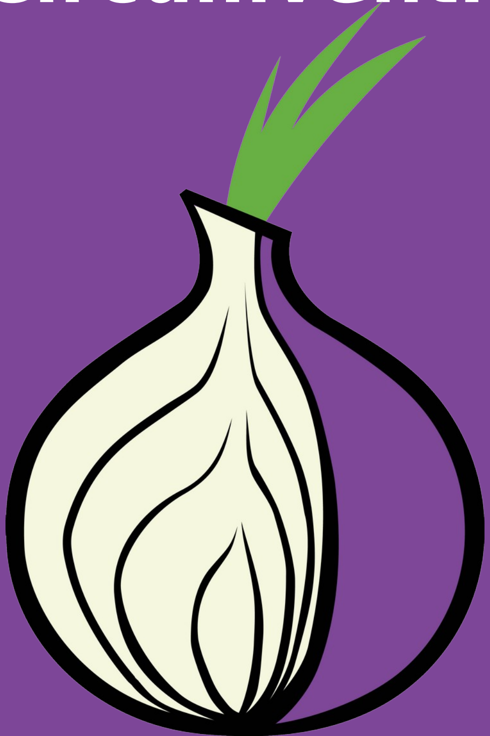
Detecting Tor Bridge Censorship

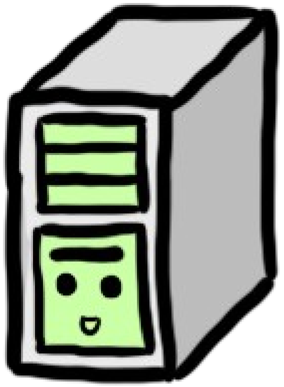
Vecna and Guy Coccimiglio

Tor as a Tool for Censorship Circumvention

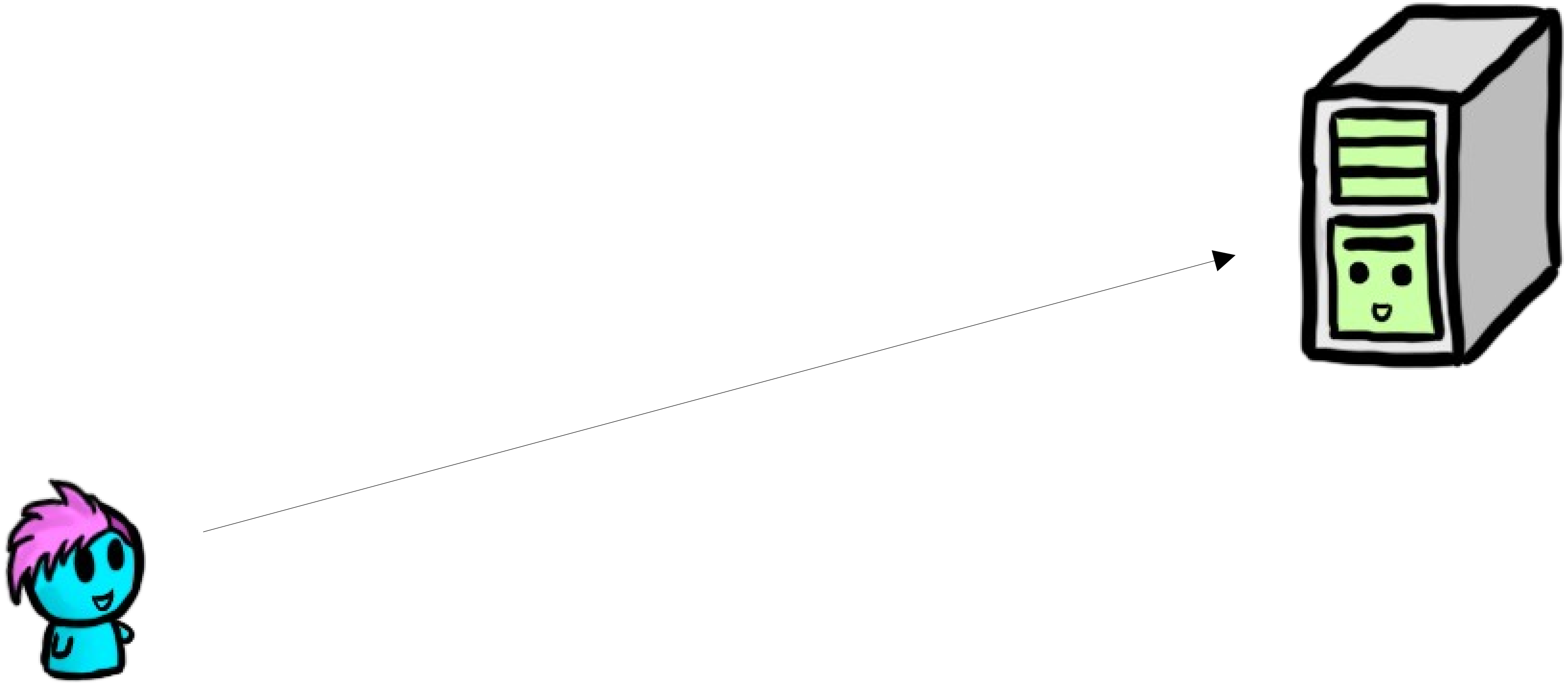


Internet Censorship

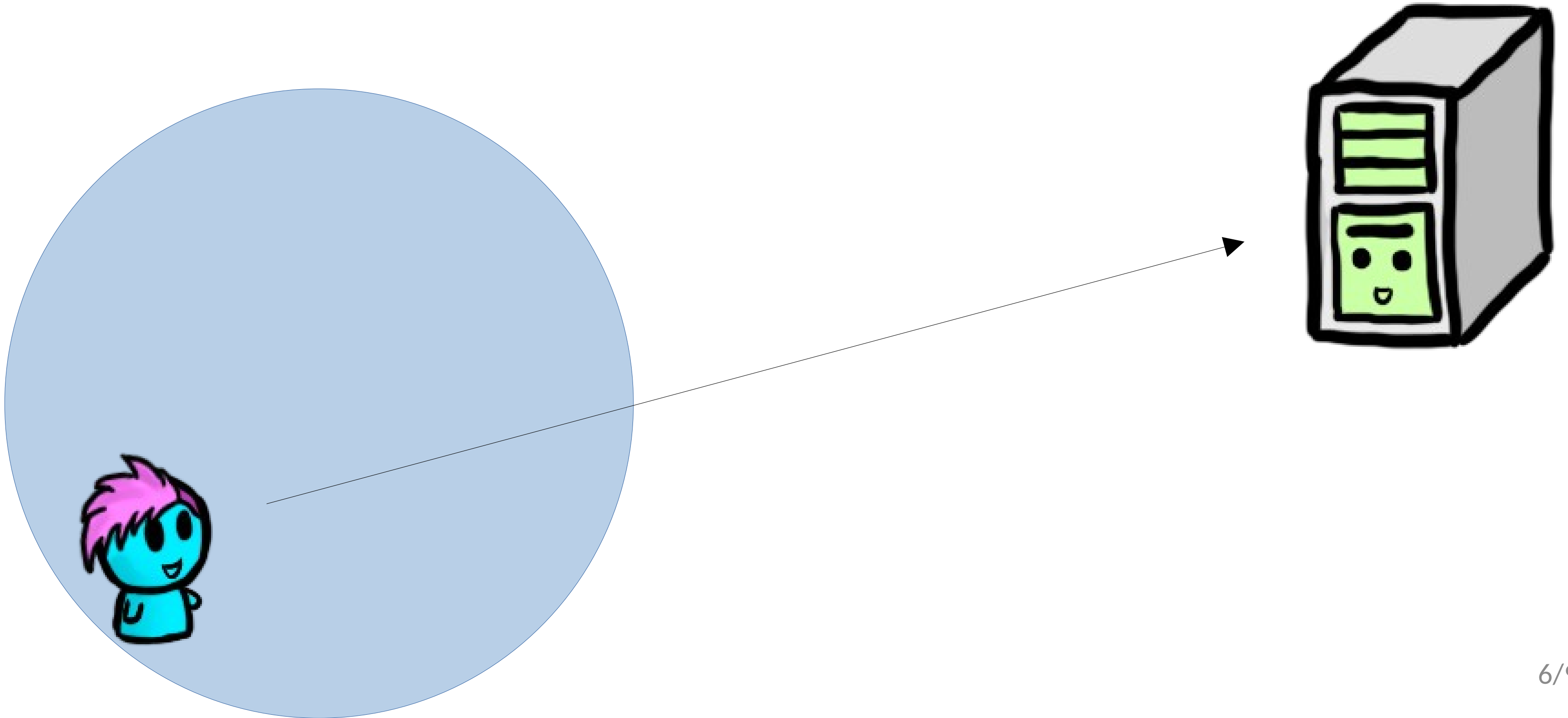
Internet Censorship



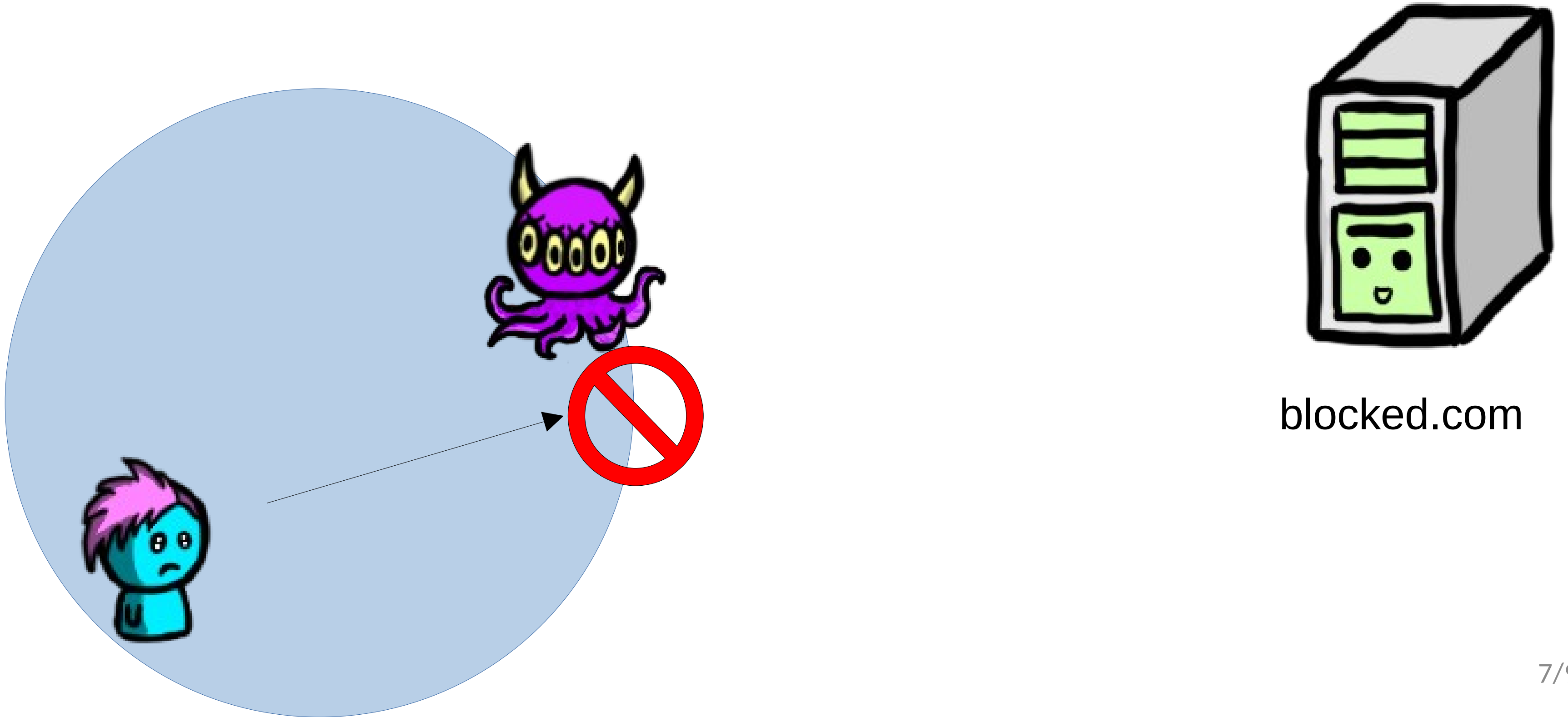
Internet Censorship



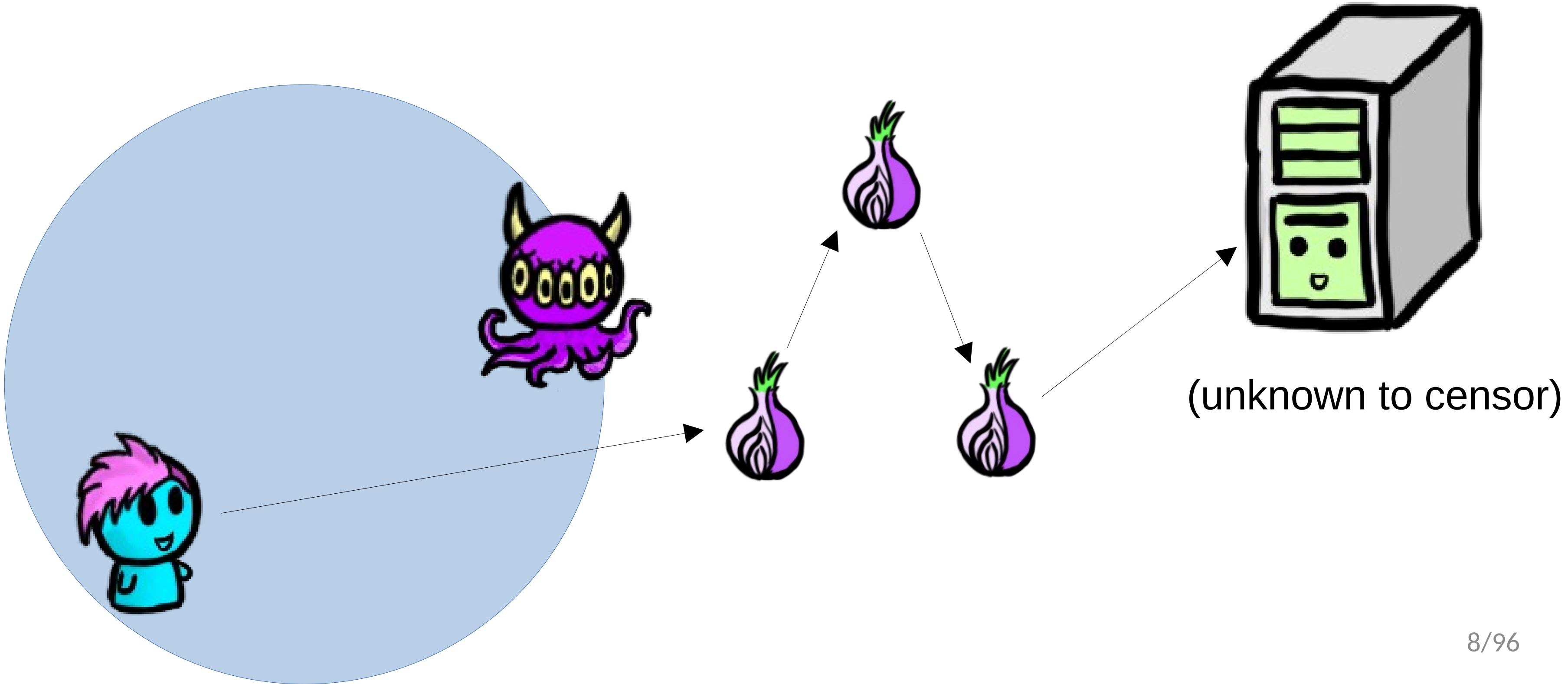
Internet Censorship



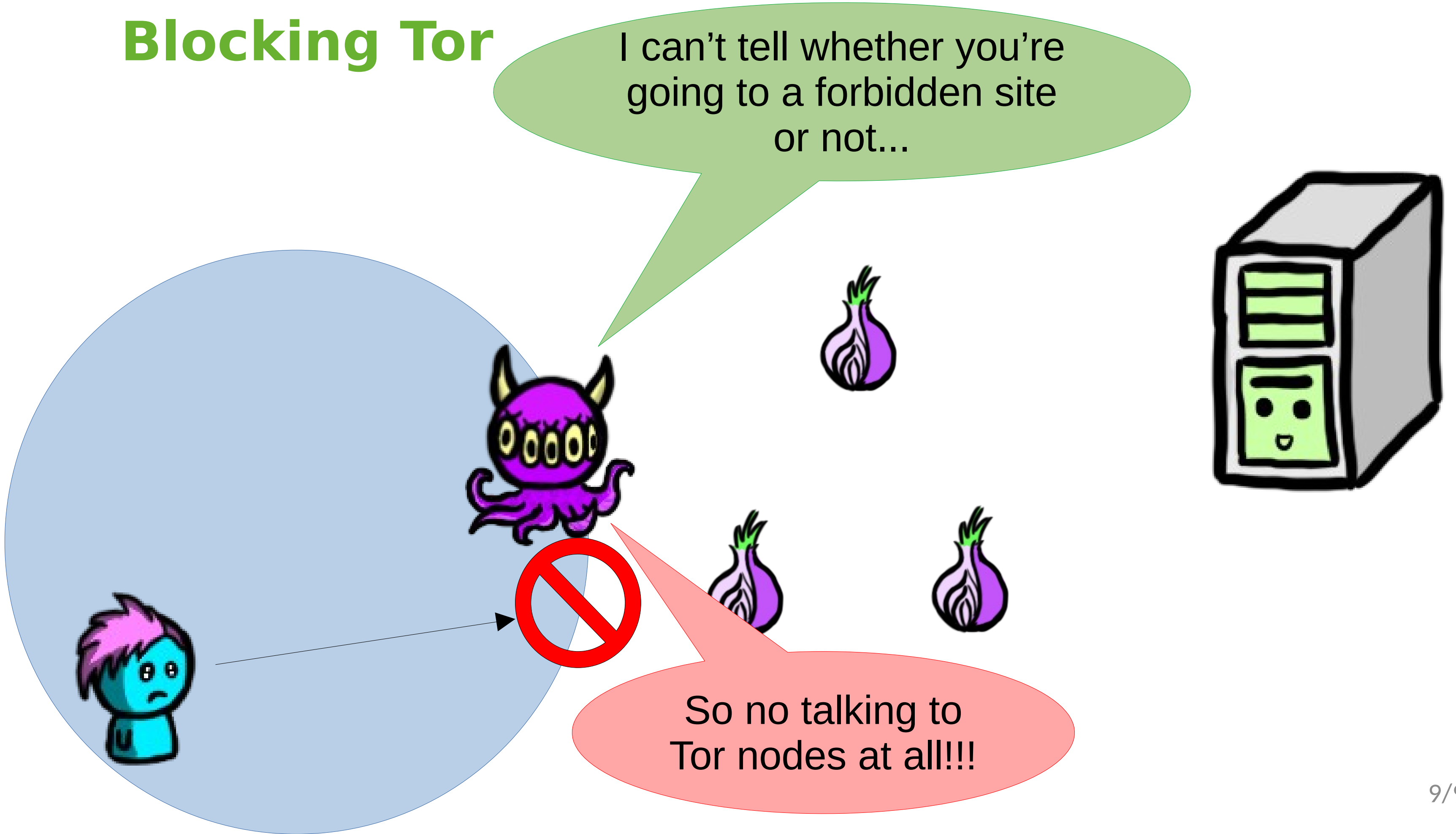
Internet Censorship



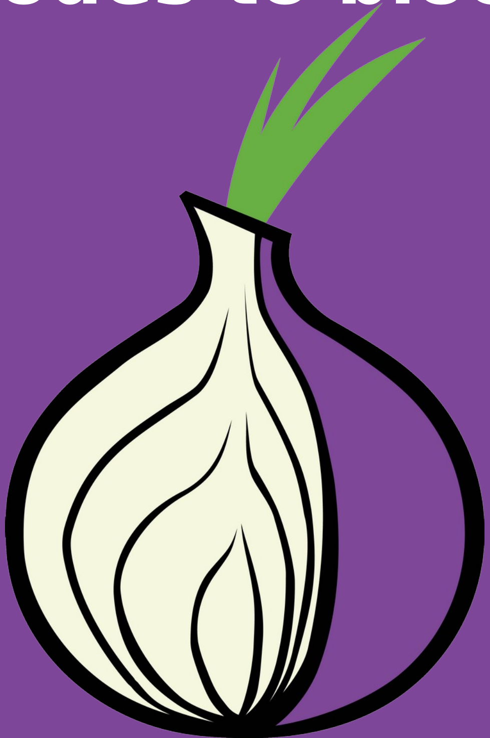
Tor



Blocking Tor

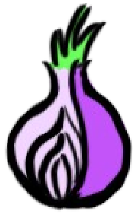

I can’t tell whether you’re going to a forbidden site or not...
So no talking to Tor nodes at all!!!

How does the censor know which nodes to block?



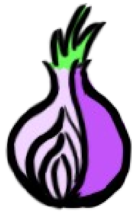
Tor node directory

seele



Address: 104.53.221.159:9001
Fingerprint: 000A10D43011E…
Flags: Fast, HSDir, Running, Stable, V2Dir, Valid

freedomrunner



Address: 198.98.61.11:9001
Fingerprint: 0011F7E36734D6…
Flags: Fast, Guard, Running, Stable, V2Dir, Valid

...

Tor node directory

seele

Address: 104.53.221.159:9001
Fingerprint: 000A10D43011E…
Flags: Fast, HSDir, Running, Stable, V2Dir, Valid

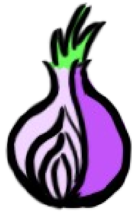
freedomrunner

Address: 198.98.61.11:9001
Fingerprint: 0011F7E36734D6…
Flags: Fast, Guard, Running, Stable, V2Dir, Valid

...

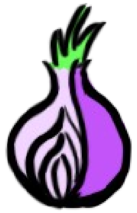
Now I know which nodes to use!

Tor node directory



seele

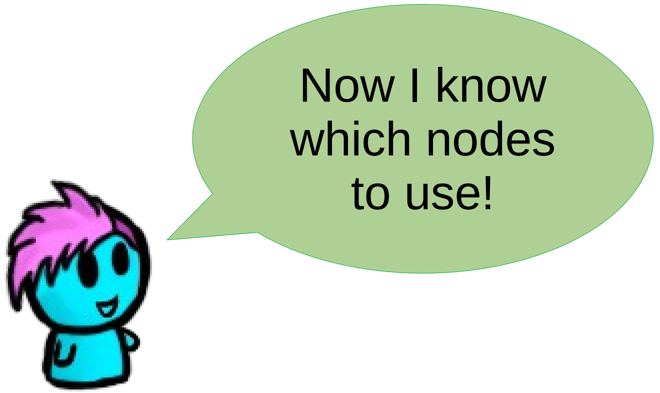
Address: 104.53.221.159:9001
Fingerprint: 000A10D43011E…
Flags: Fast, HSDir, Running, Stable, V2Dir, Valid

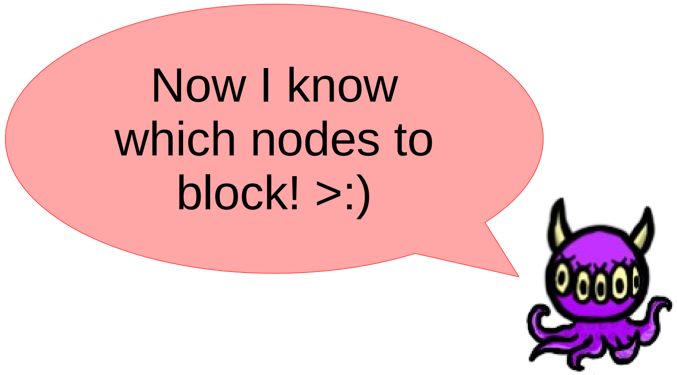


freedomrunner

Address: 198.98.61.11:9001
Fingerprint: 0011F7E36734D6…
Flags: Fast, Guard, Running, Stable, V2Dir, Valid

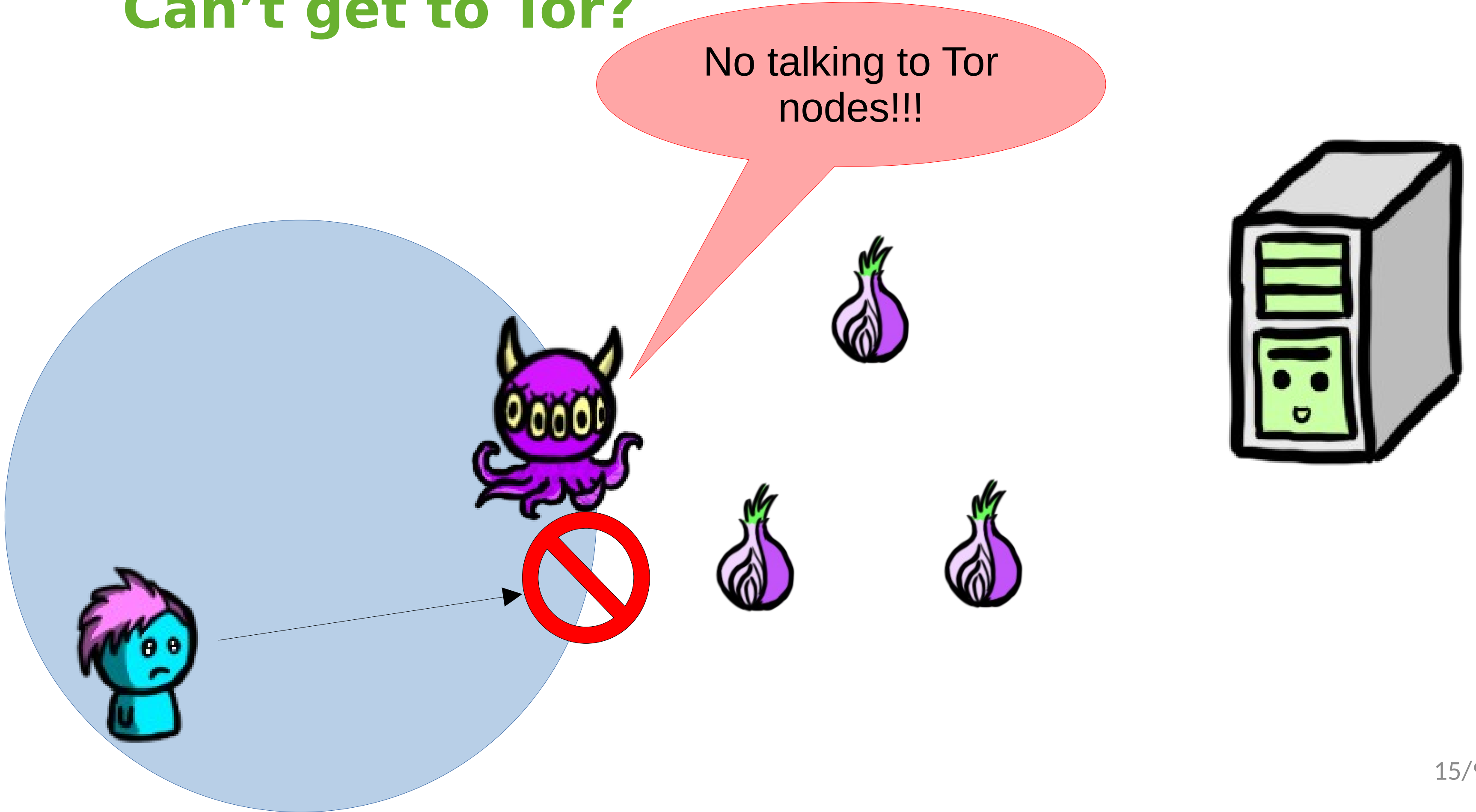
...



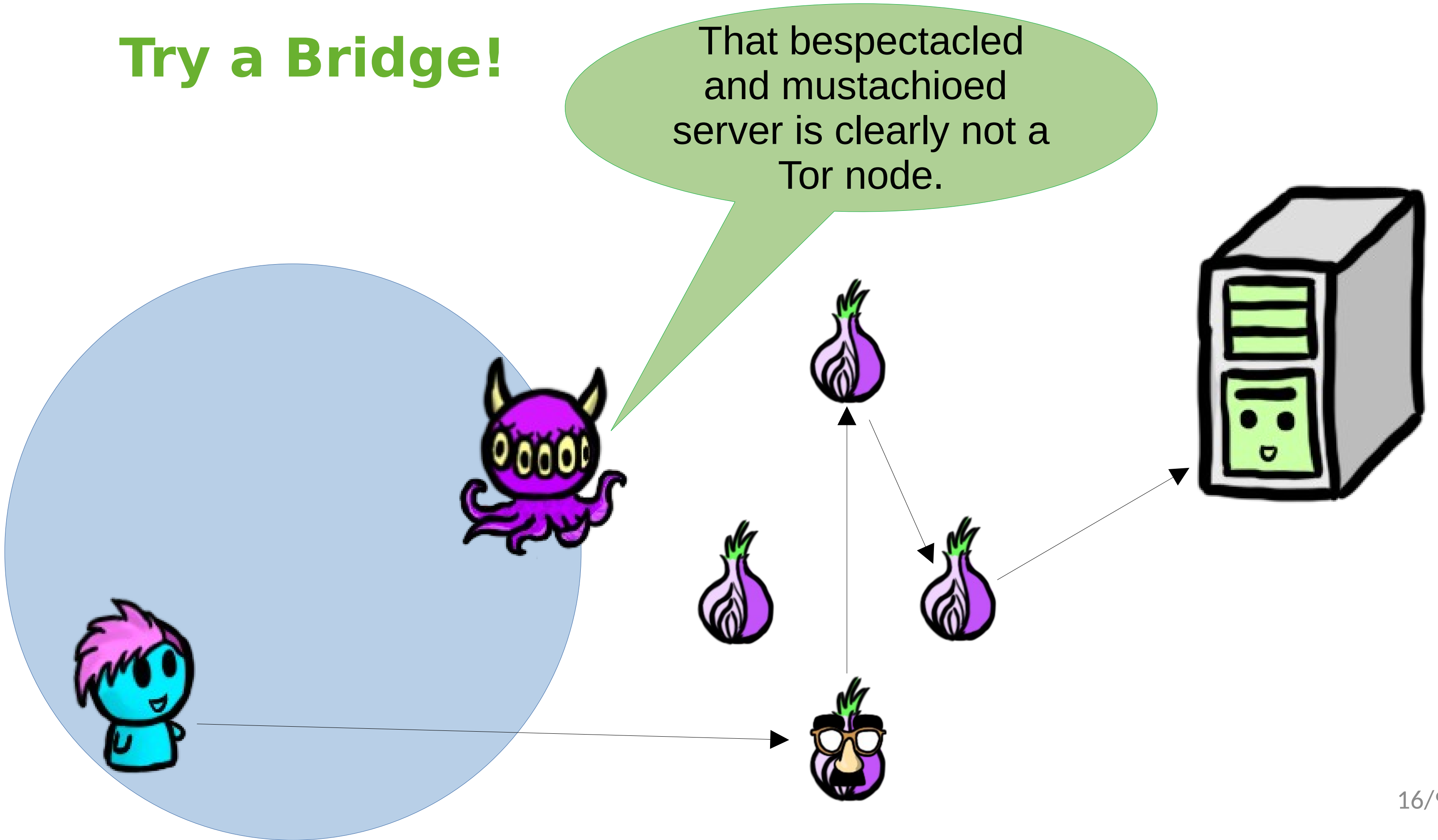


Bridges

Can’t get to Tor?

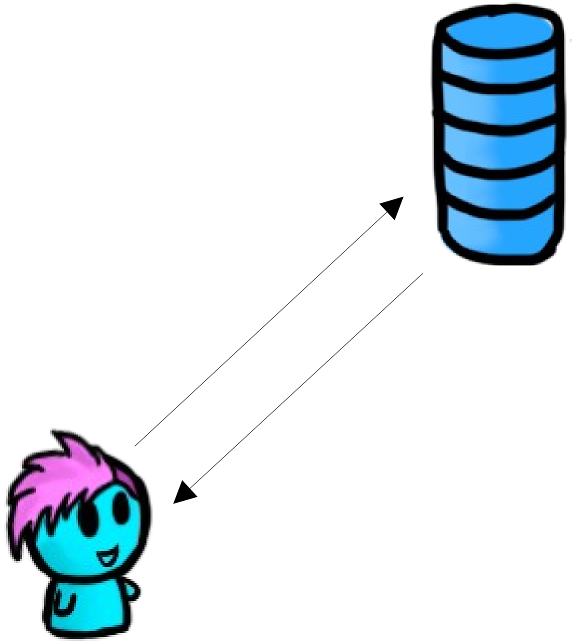


Try a Bridge!


That bespectacled and mustachioed server is clearly not a Tor node.

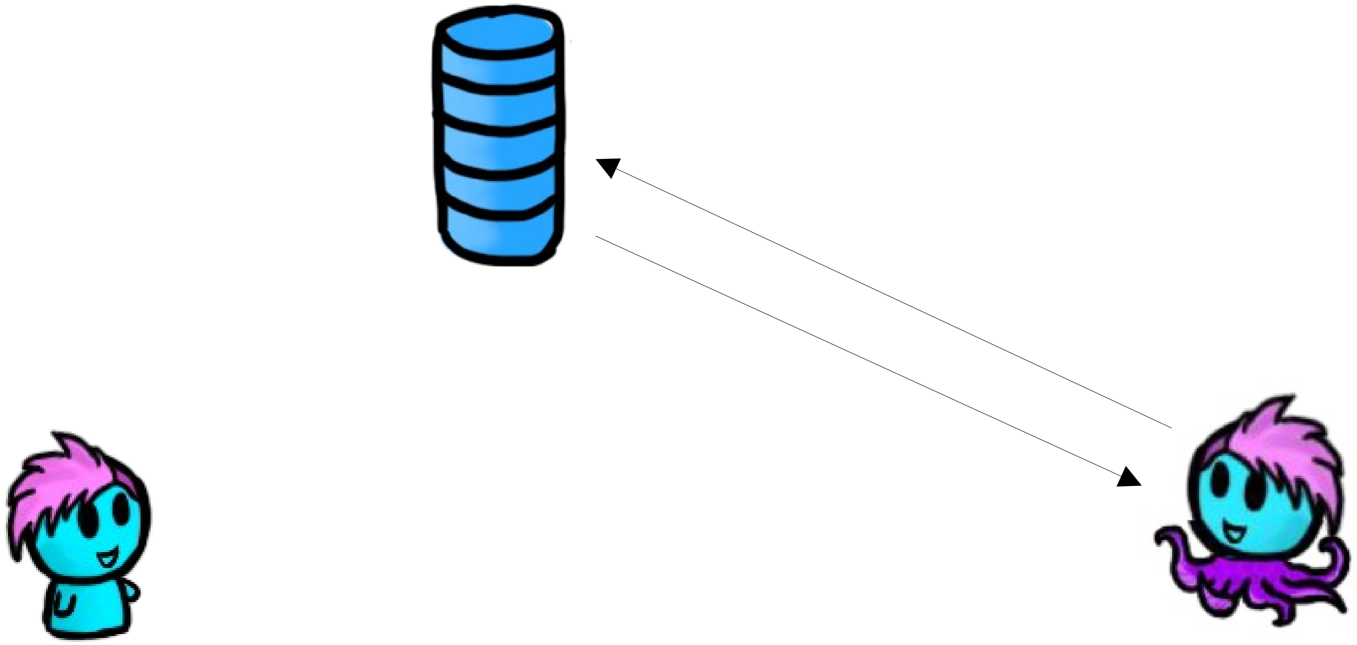
How Do Users Learn About Bridges?

- BridgeDB (email/HTTPS)



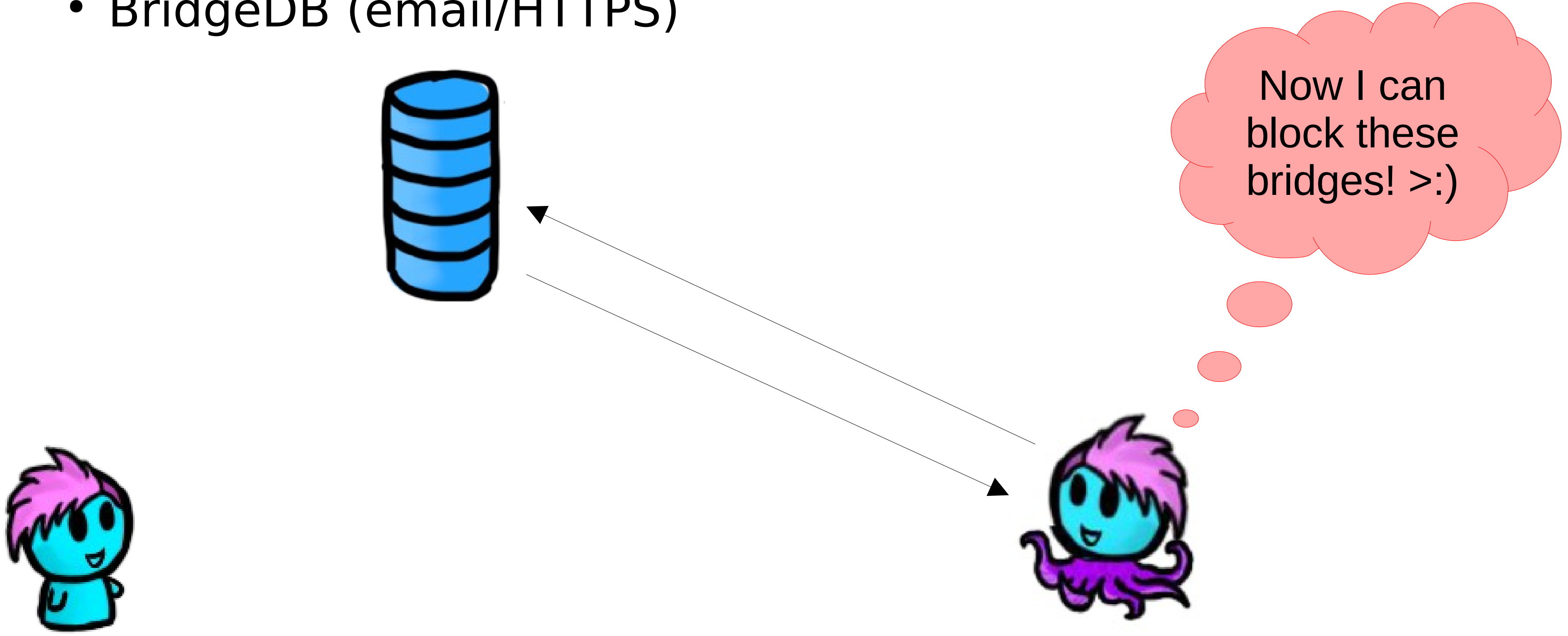
How Do Censors Learn About Bridges?

- BridgeDB (email/HTTPS)



How Do Censors Learn About Bridges?

- BridgeDB (email/HTTPS)



New Bridge Distribution Systems

- New systems
  - Punish users when bridges are blocked
  - Reward users when bridges are not blocked

New Bridge Distribution Systems

- New systems
  - Punish users when bridges are blocked
  - Reward users when bridges are not blocked

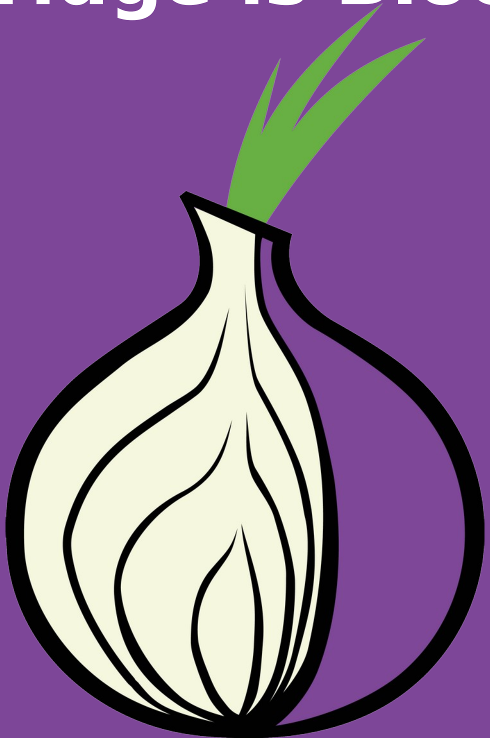
New Bridge Distribution Systems

- New systems
  - Punish users when bridges are blocked
  - Reward users when bridges are not blocked

New Bridge Distribution Systems

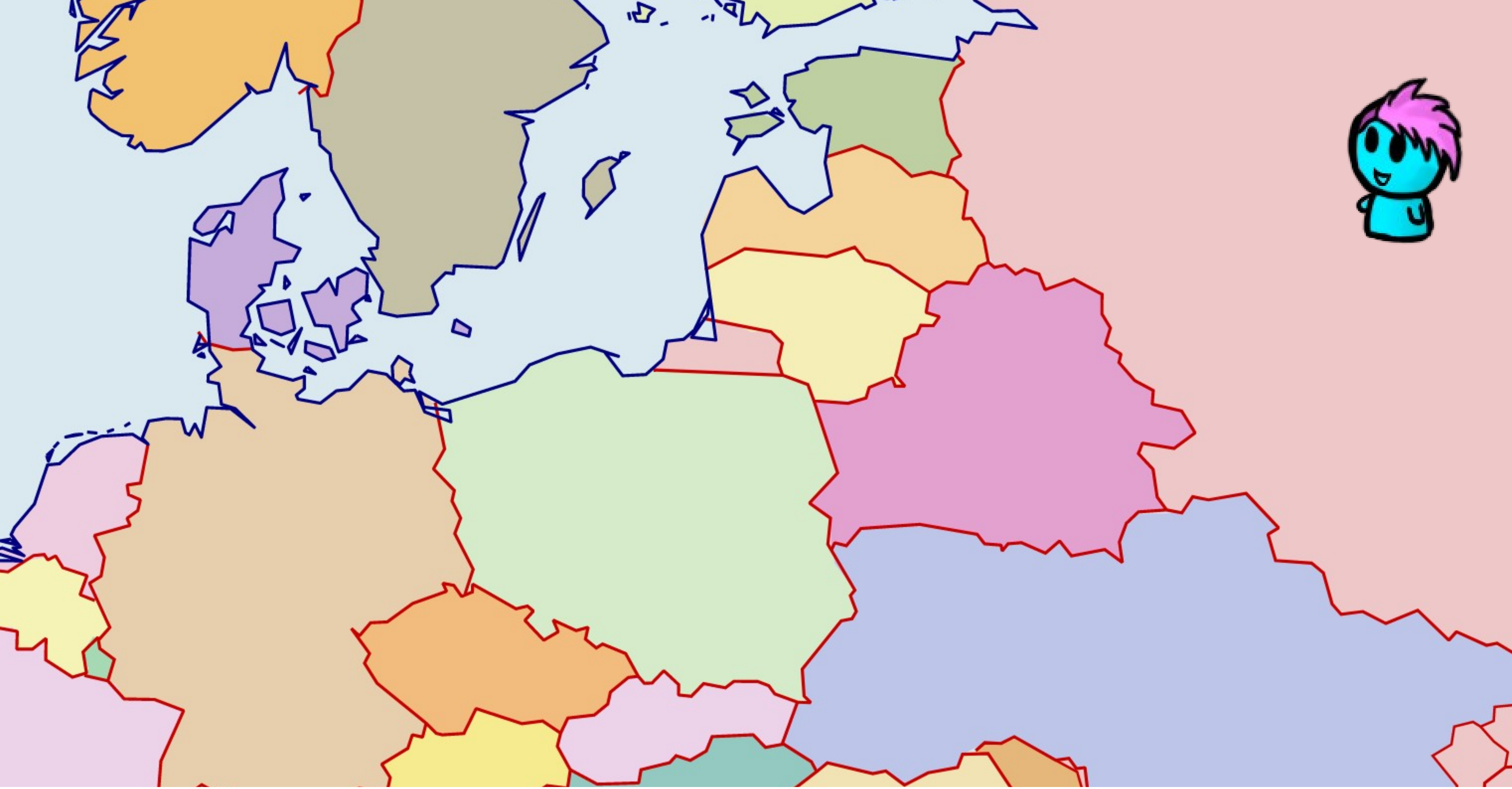
- New systems
  - Punish users when bridges are blocked
  - Reward users when bridges are not blocked

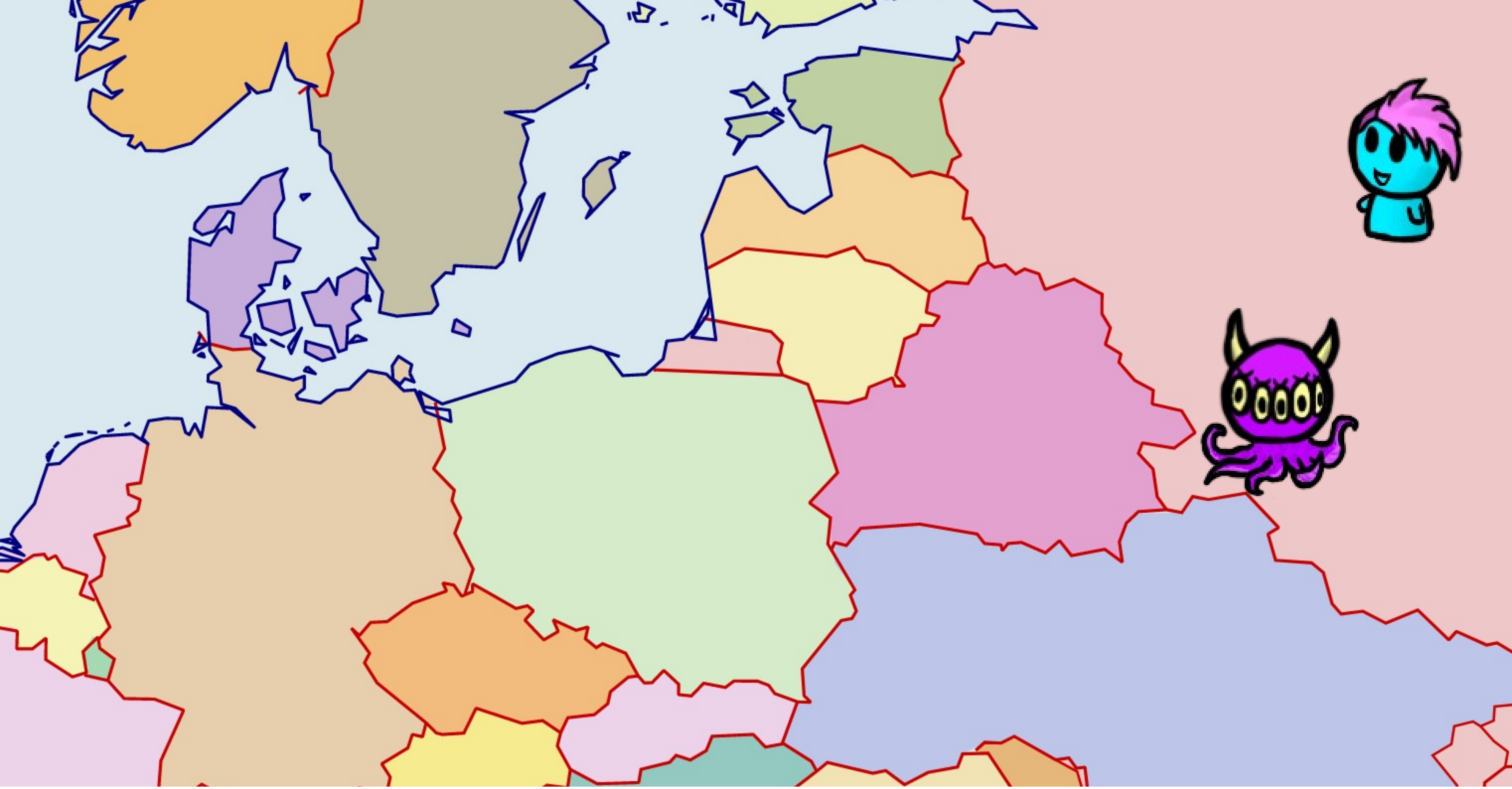
How Do We Know Whether a Bridge is Blocked?

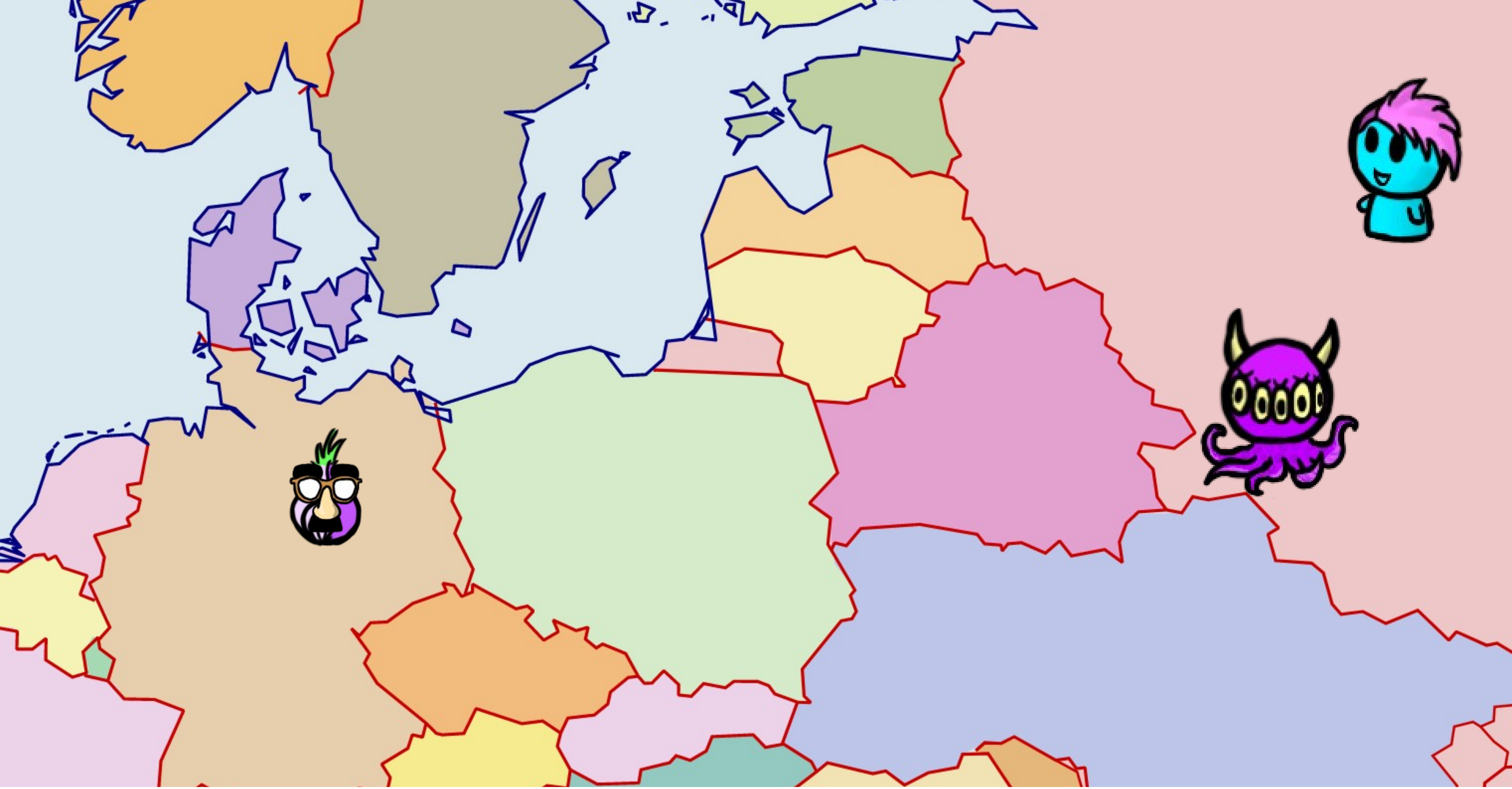


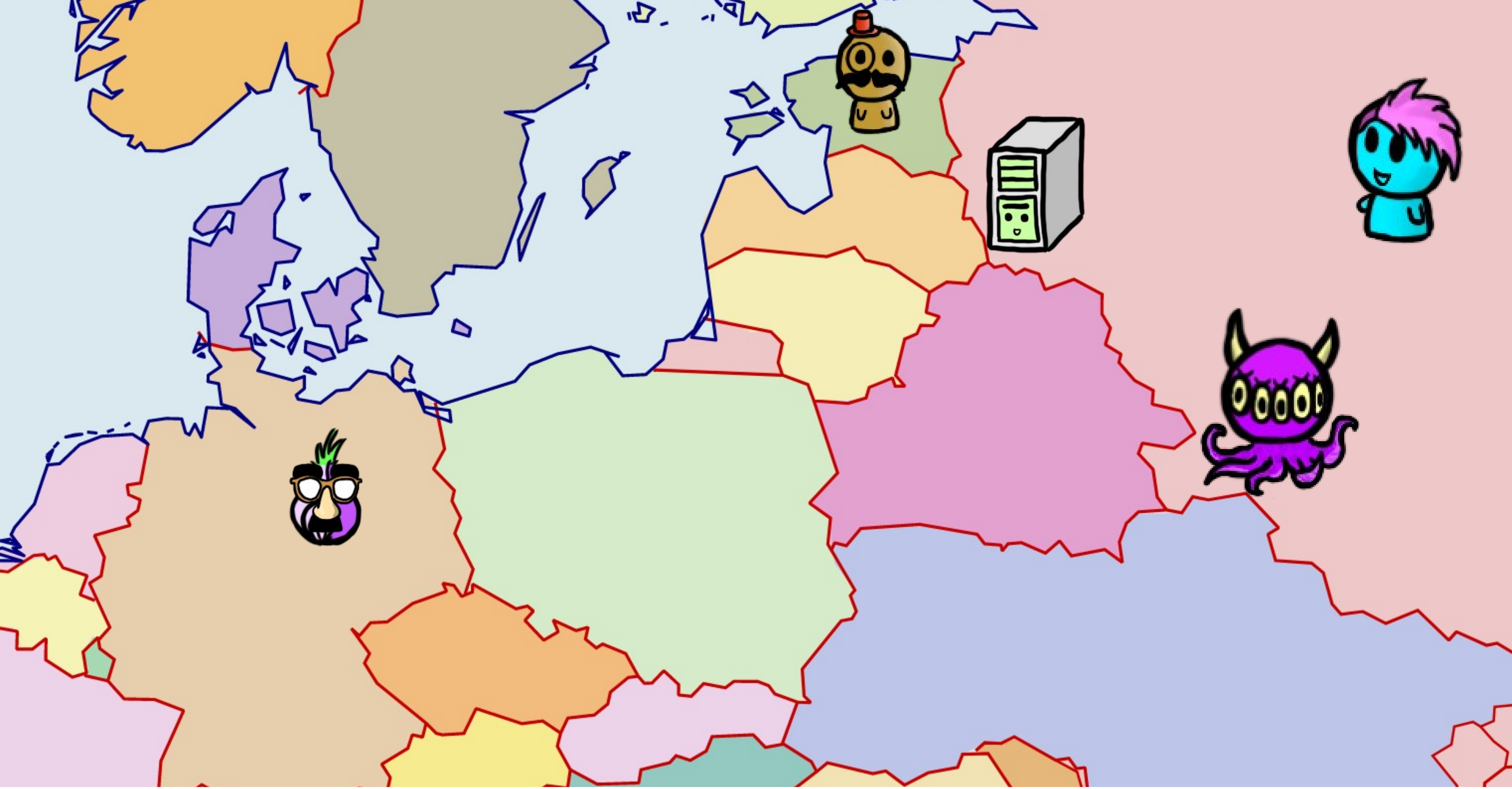
Detecting Bridge Blocking

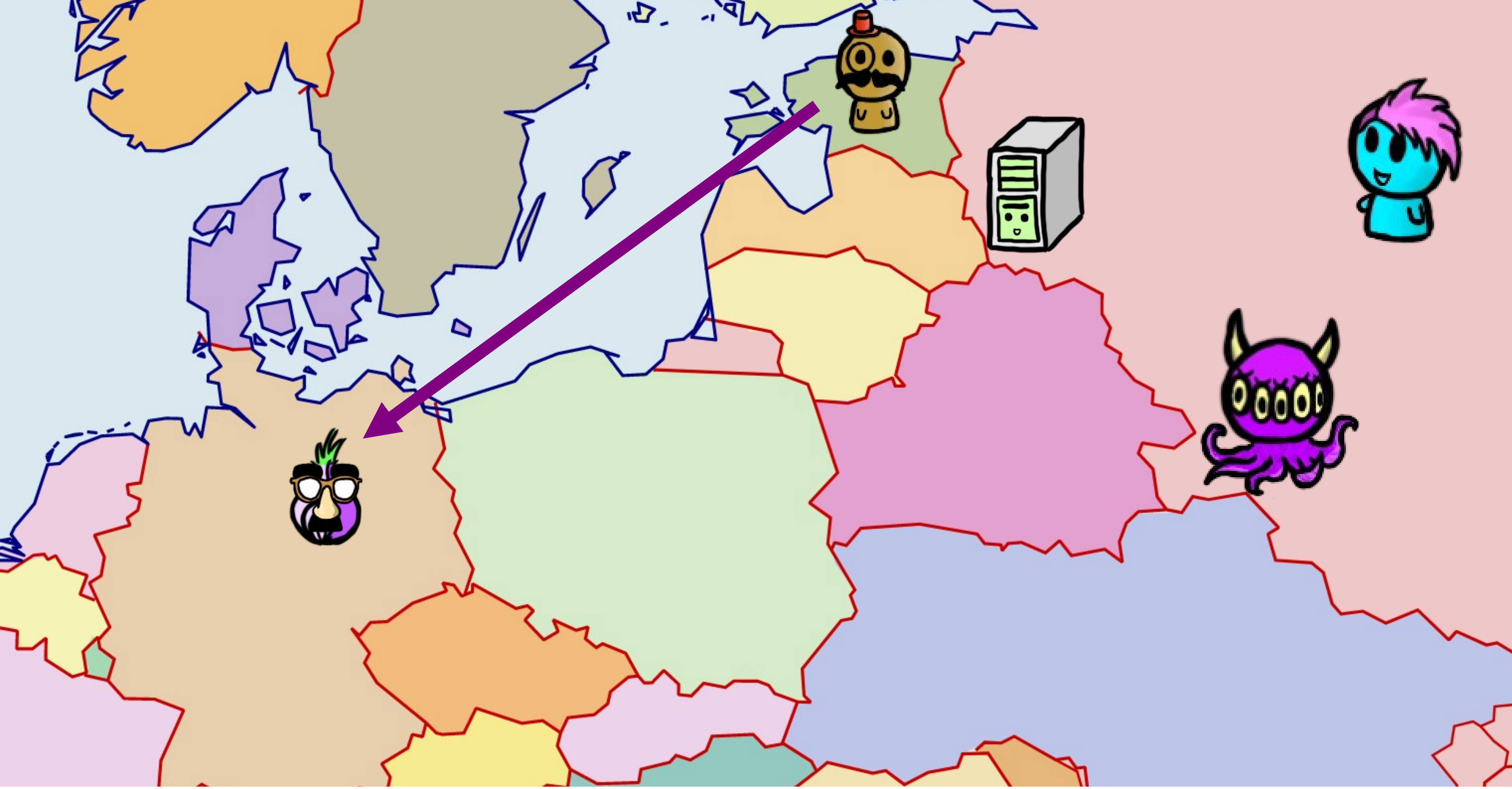
- **Probing bridges**
- User reports
- Bridge stats

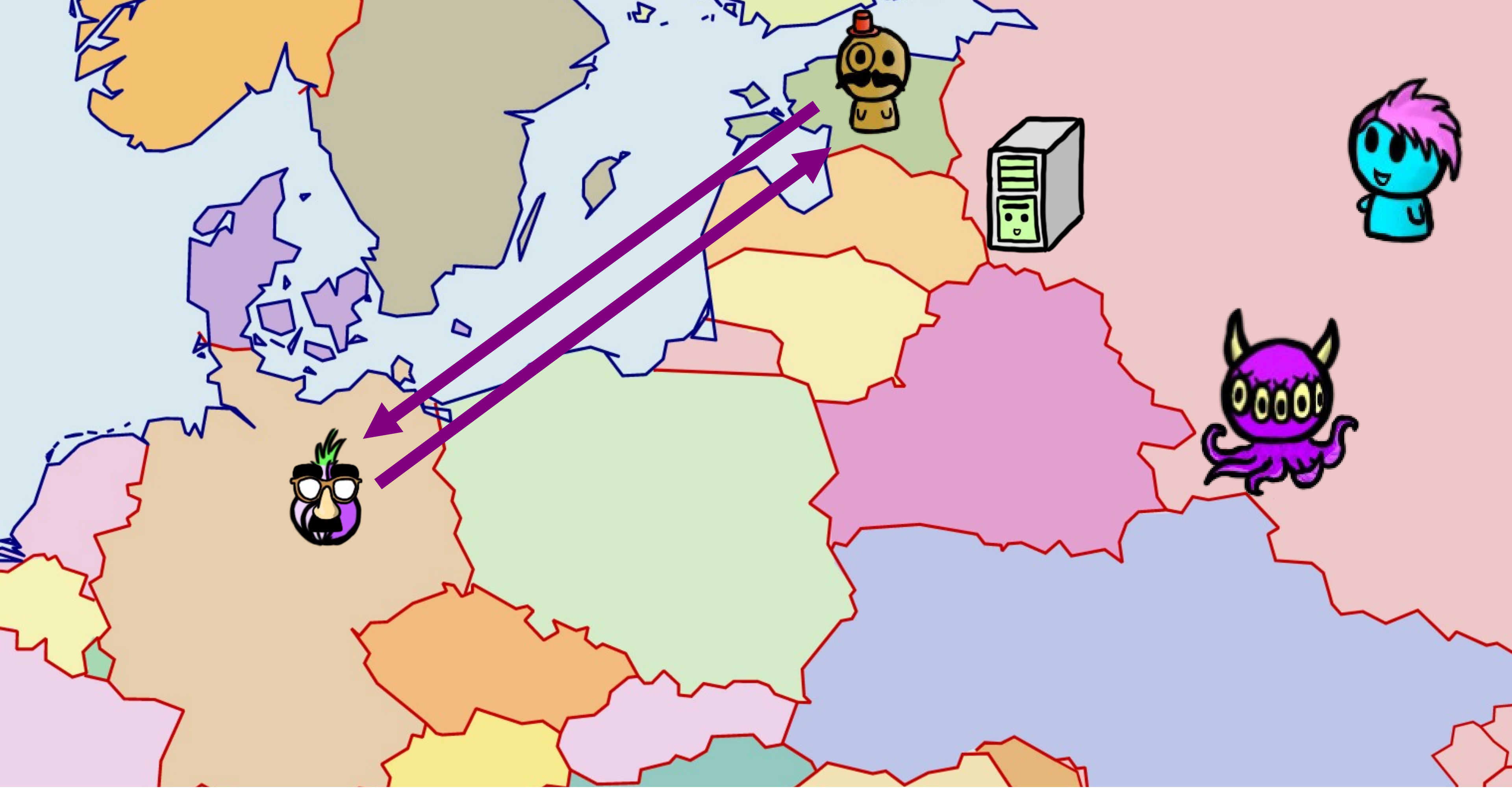


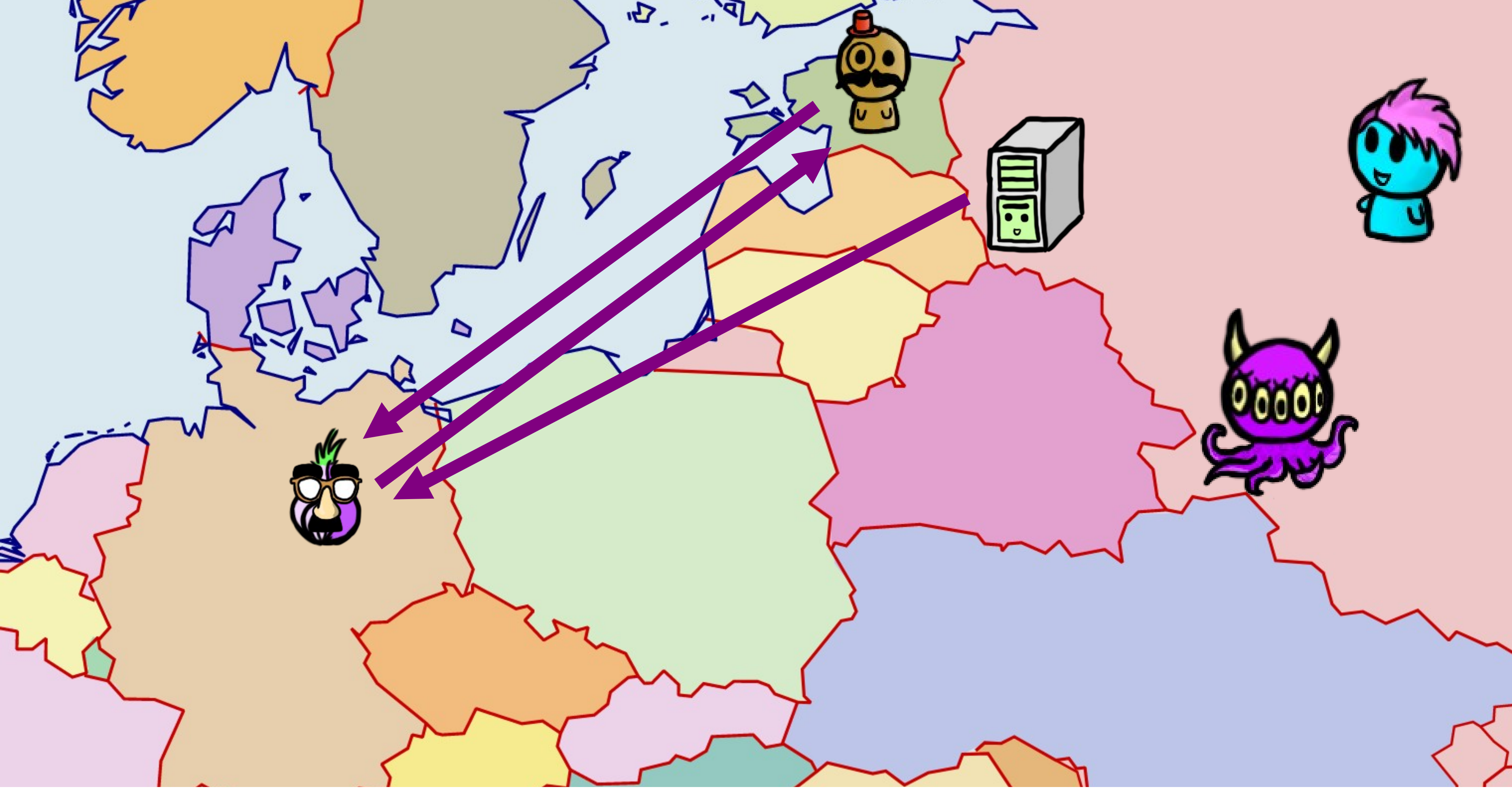


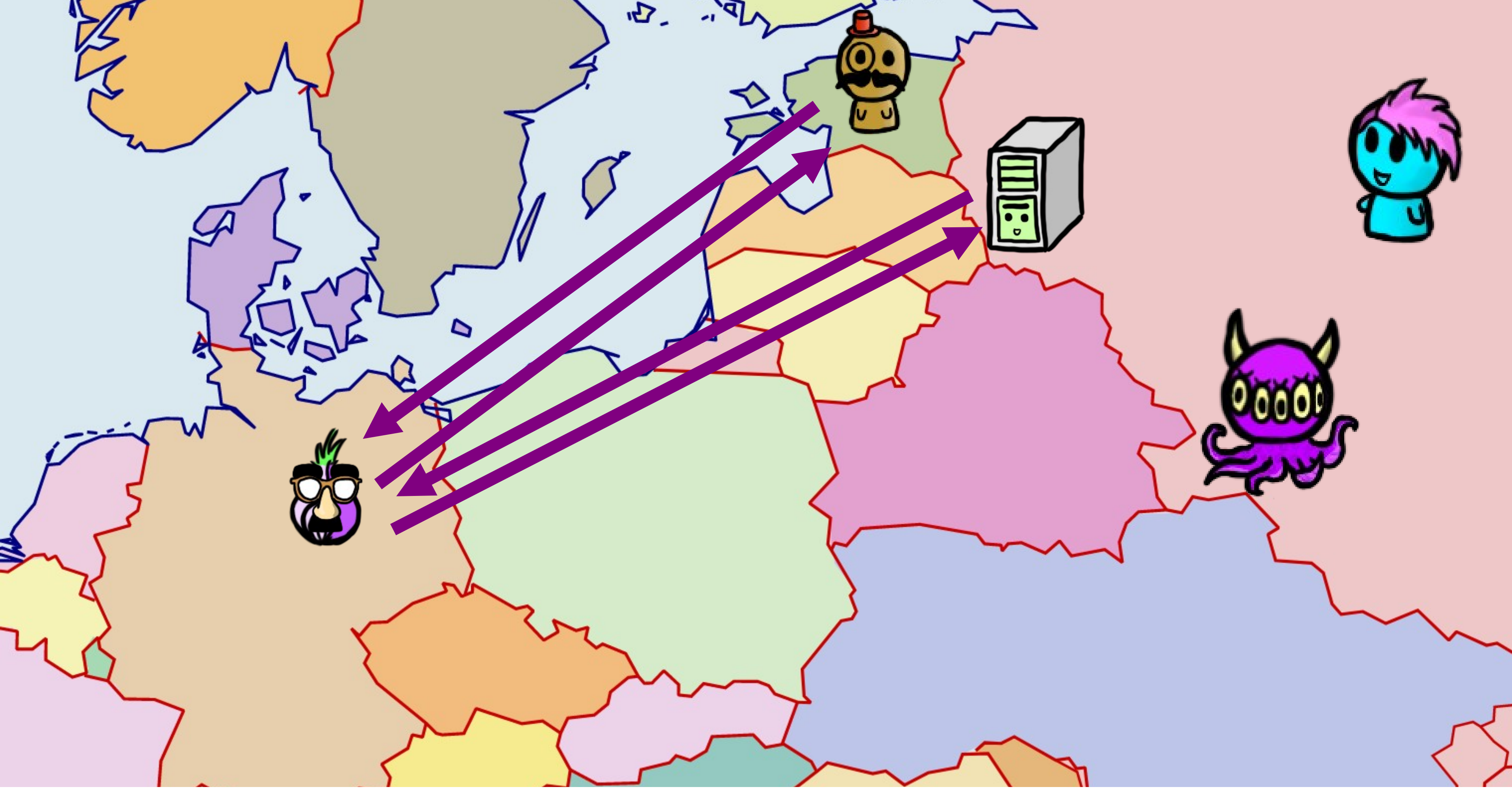


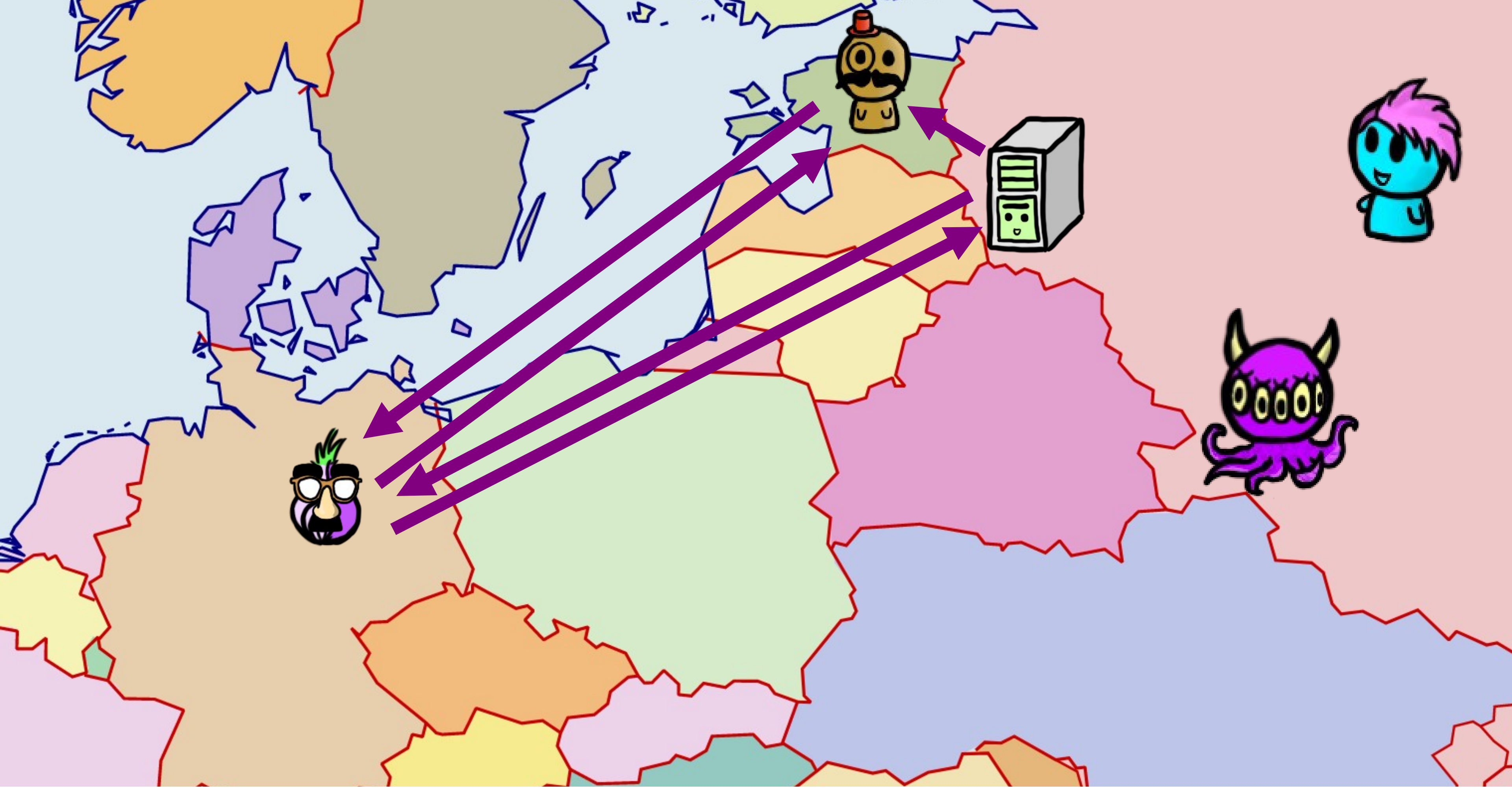






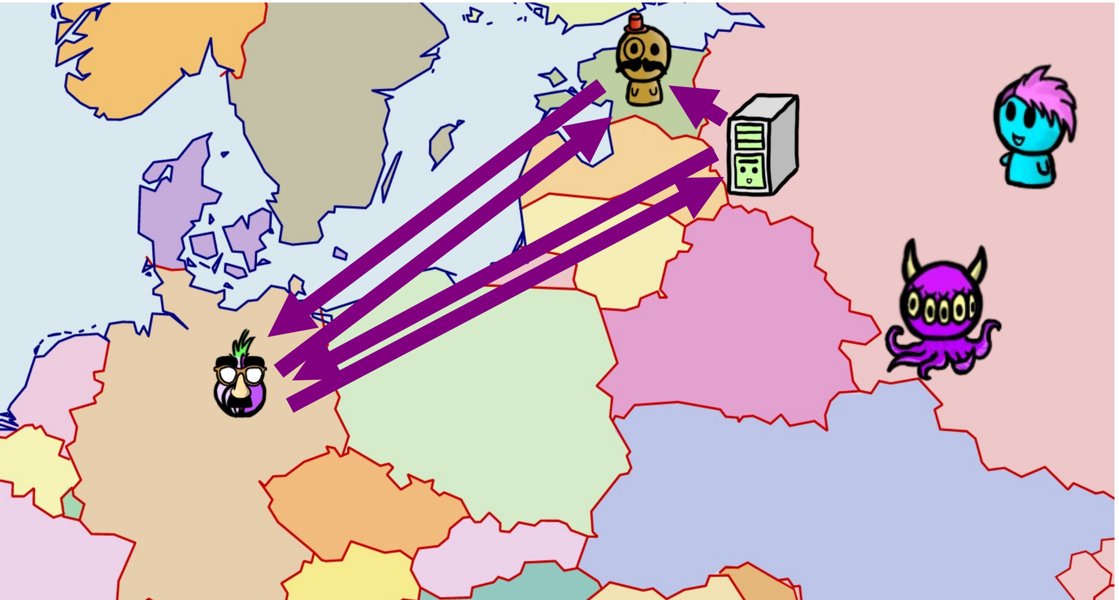






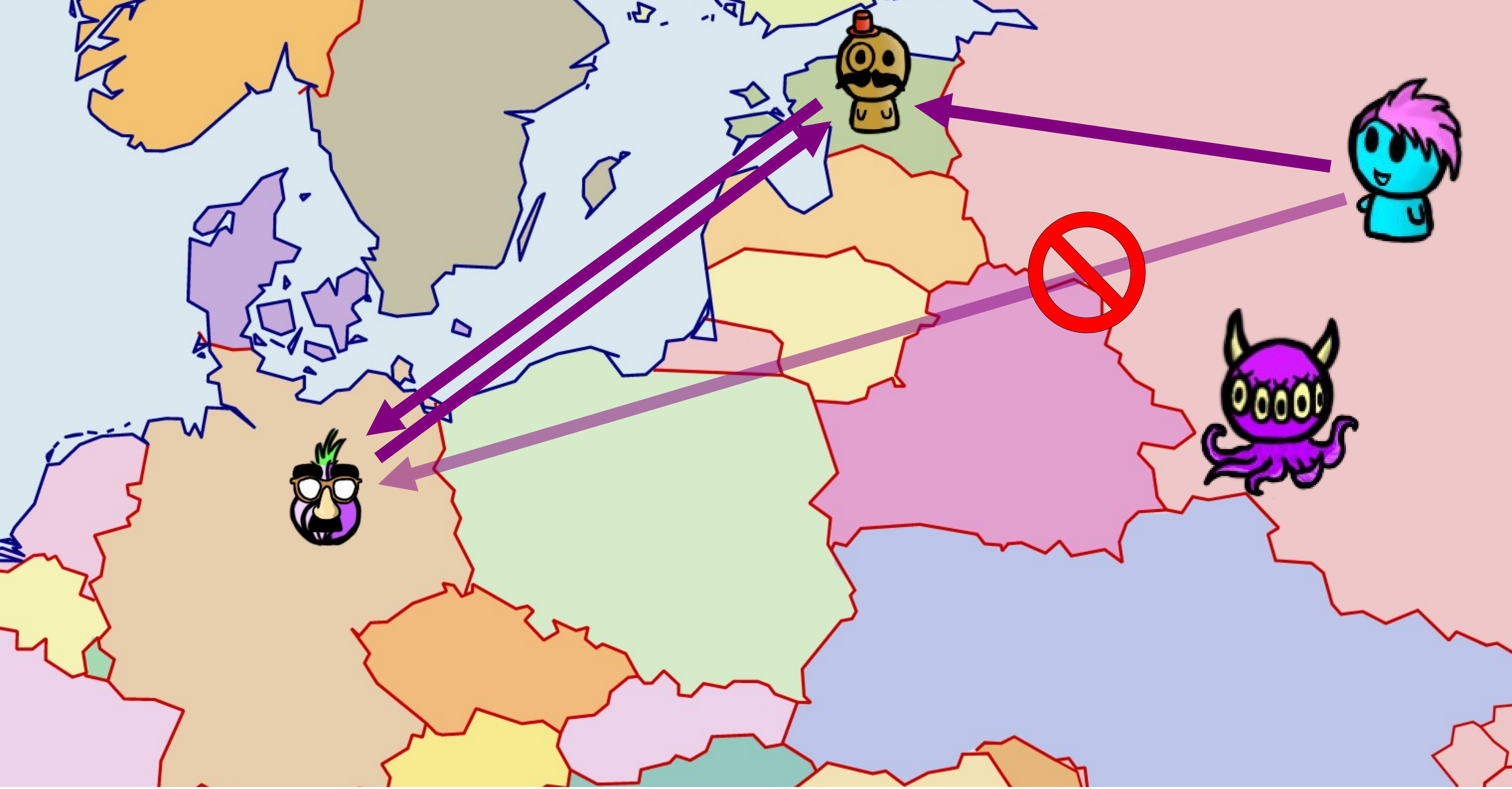
Concerns with Probing

- May attract attention of censor
- Balancing accuracy with safety



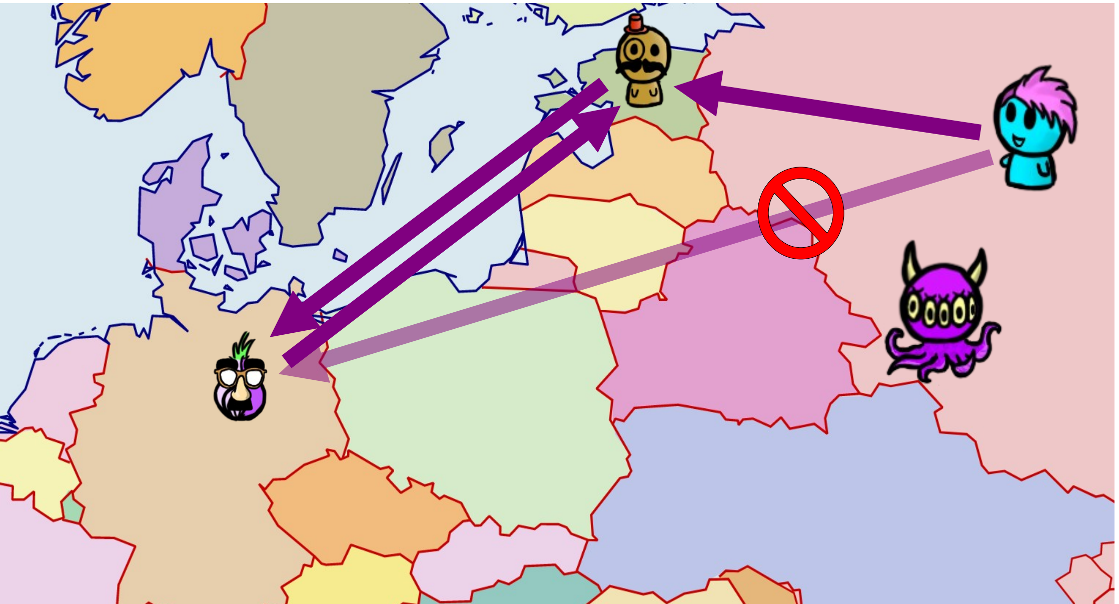
Detecting Bridge Blocking

- Probing bridges
- **User reports**
- Bridge stats



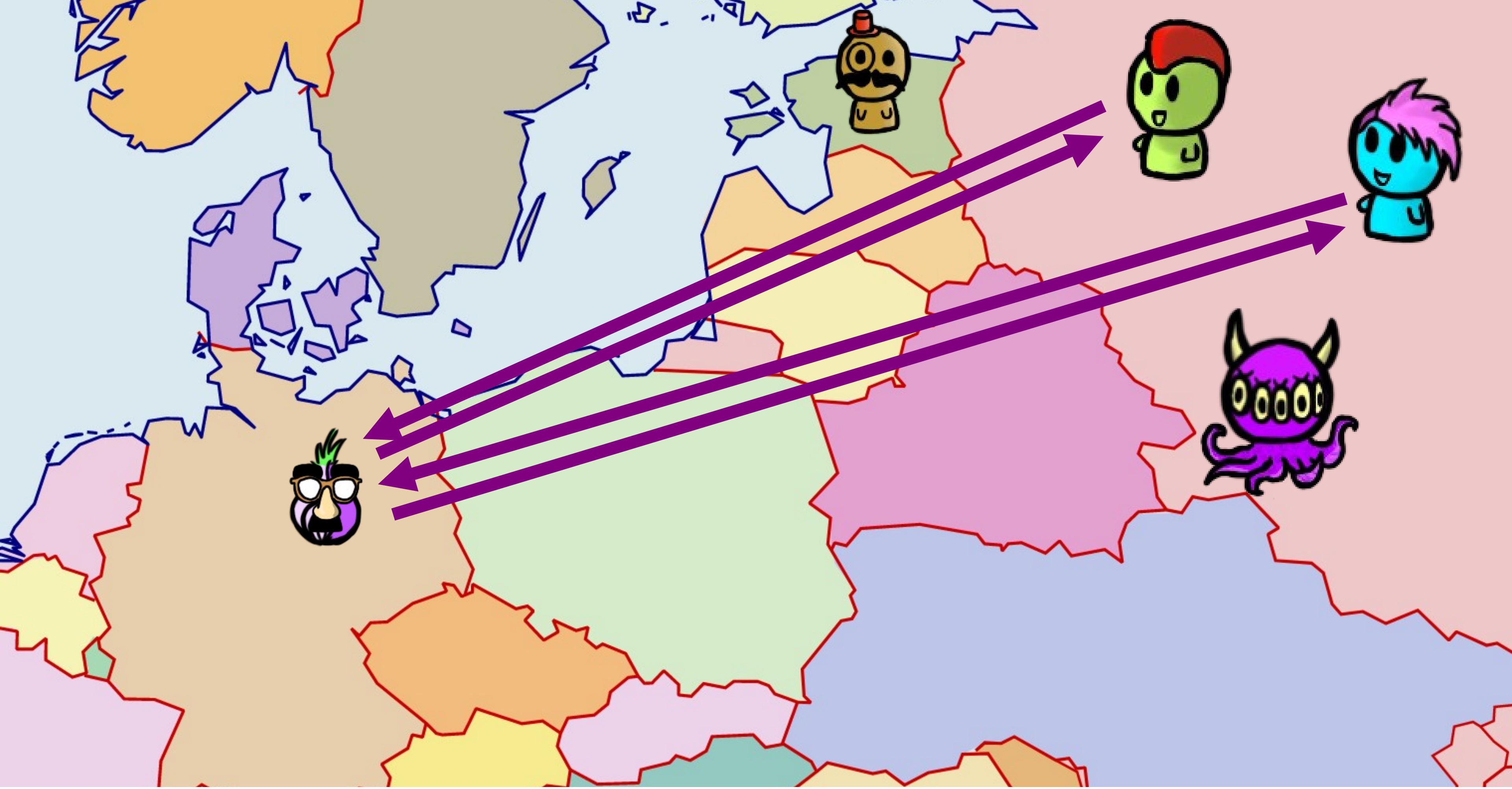
Concerns with User Reports

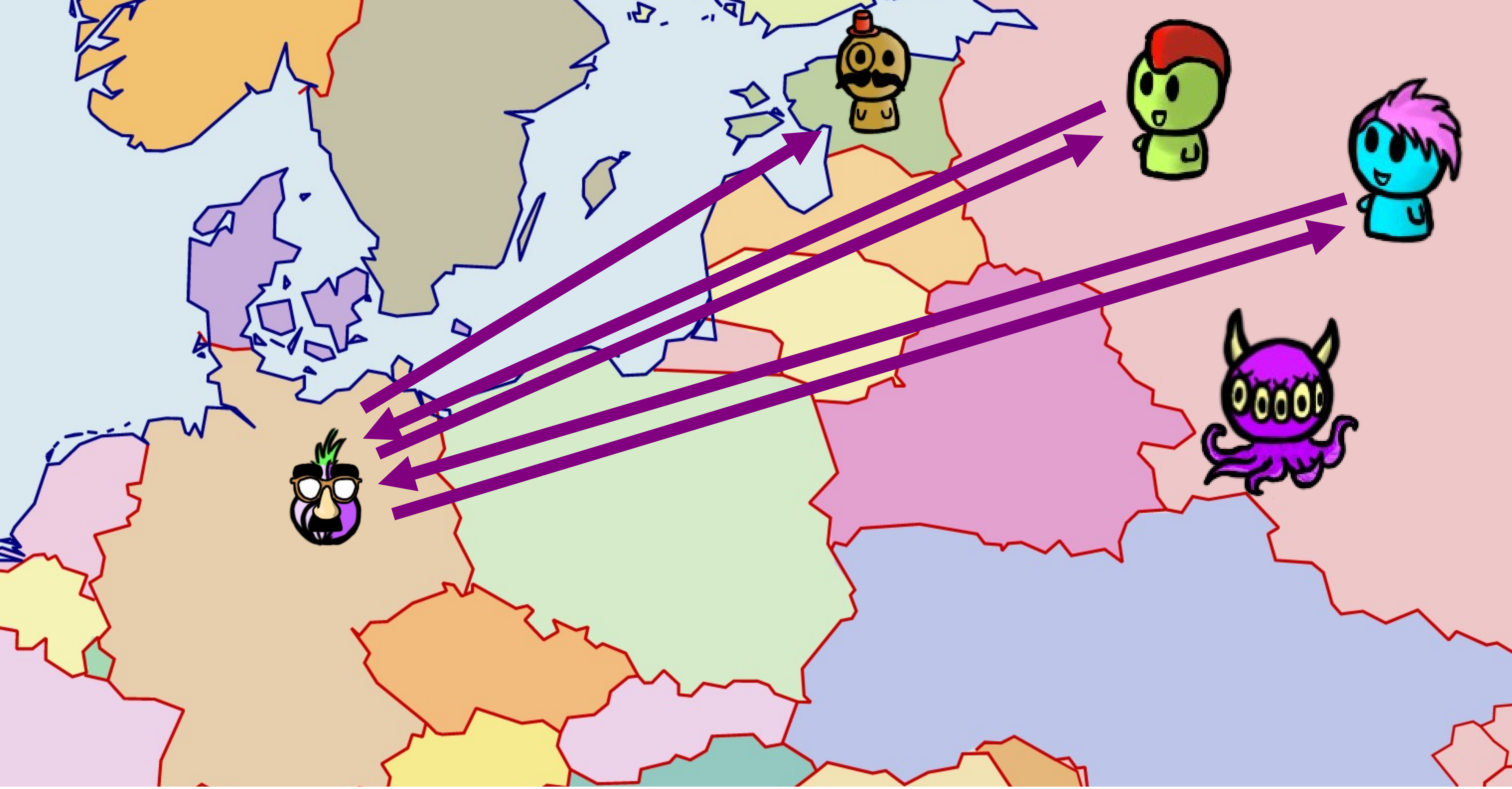
- Inaccurate reports
- Malicious reports?

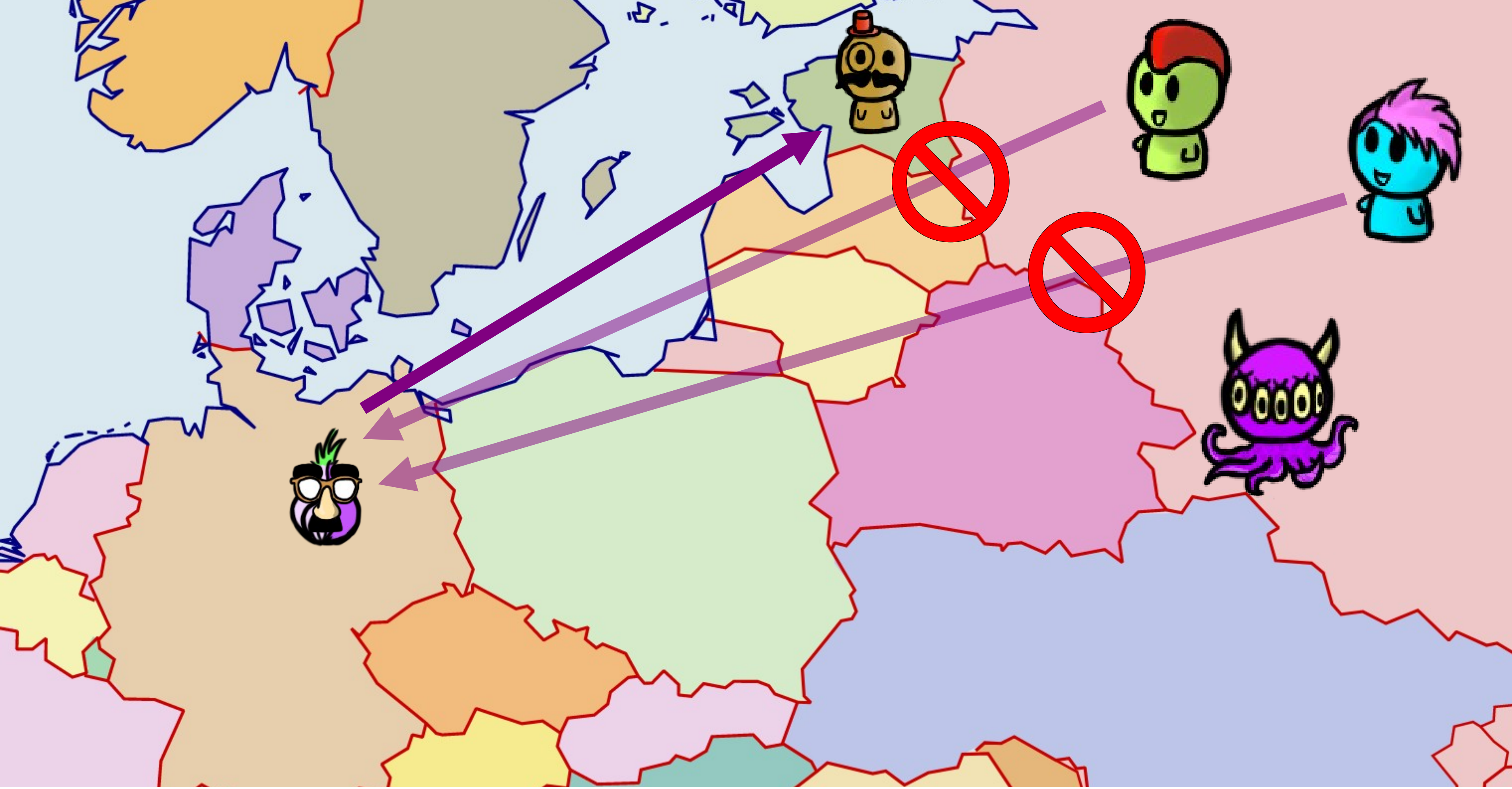


Detecting Bridge Blocking

- Probing bridges
- User reports
- **Bridge stats**







Concerns with Bridge Stats

- Relies on GeoIP information
- Hard to estimate accurately

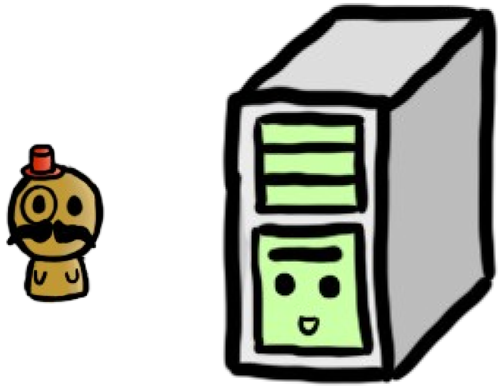
Detecting Blocked Bridges

Blockage Detection Algorithm

- 2 Phase algorithm
- Phase 1 – user reports + bridge usage statistics
  - Output: confidence that this bridge is blocked
- Phase 2 – probing
  - Output: is this bridge blocked

Blockage Detection Architecture

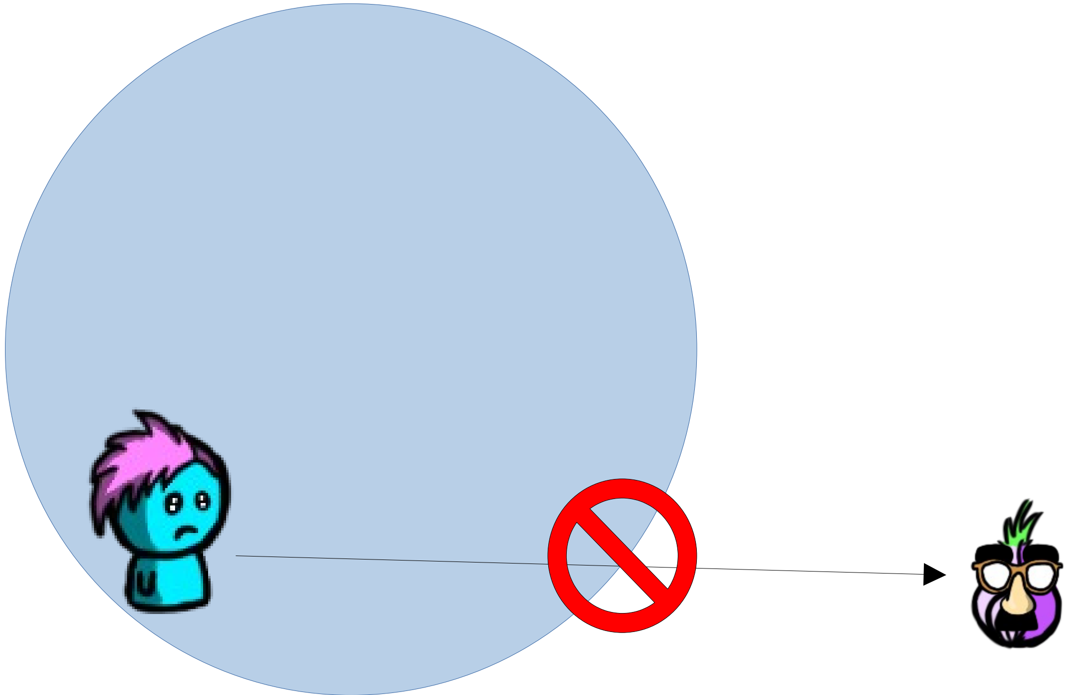
- Centralized detection server
- Periodically checks all bridges for blockages
- Could integrate with bridge authority



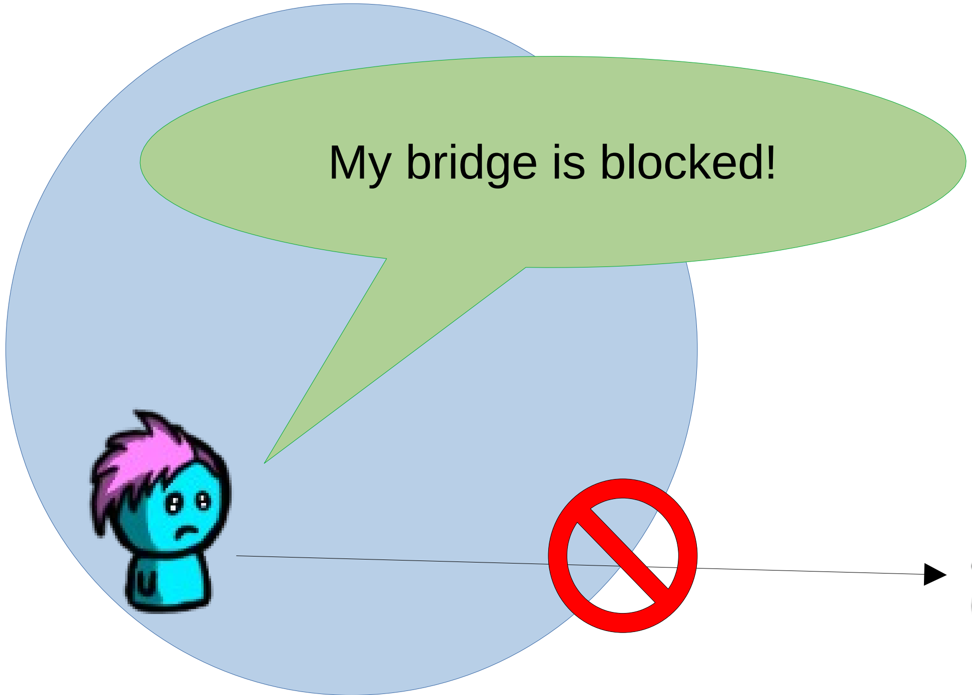
User Reports

- Data record indicating a user experienced a possible blockage
- Submitted to the detection server by users
- Minimally contains:
  - Bridge identifier
  - User’s region

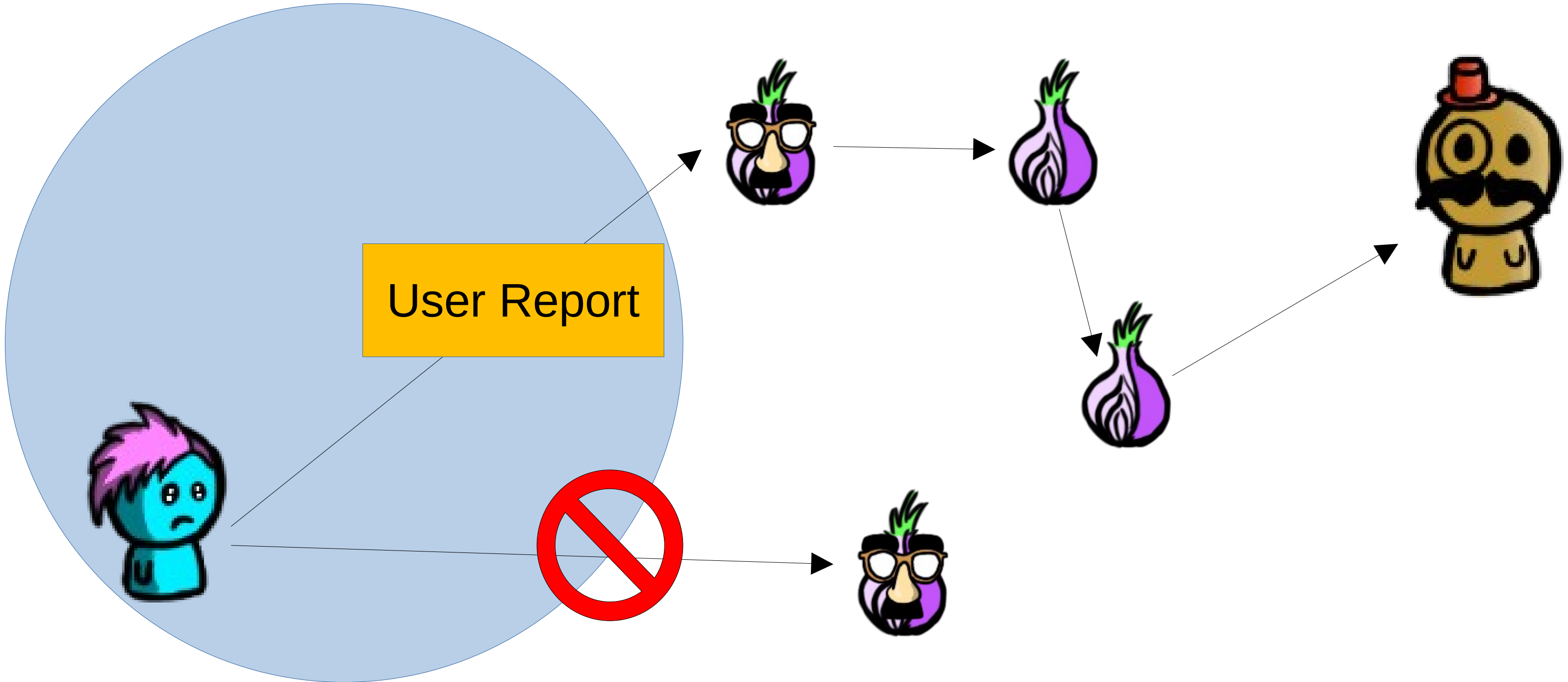
User Reports



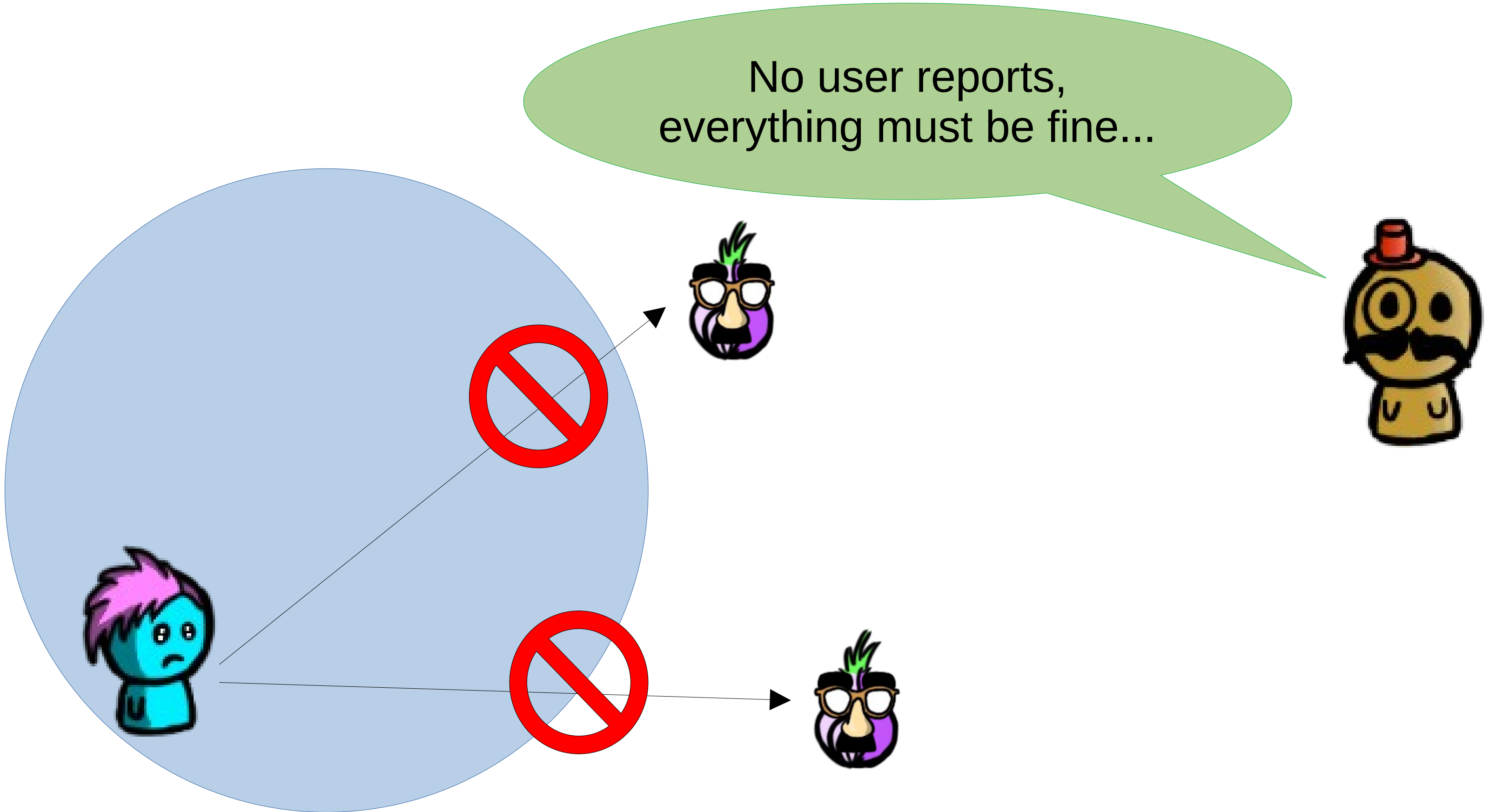
User Reports



User Reports



User Reports

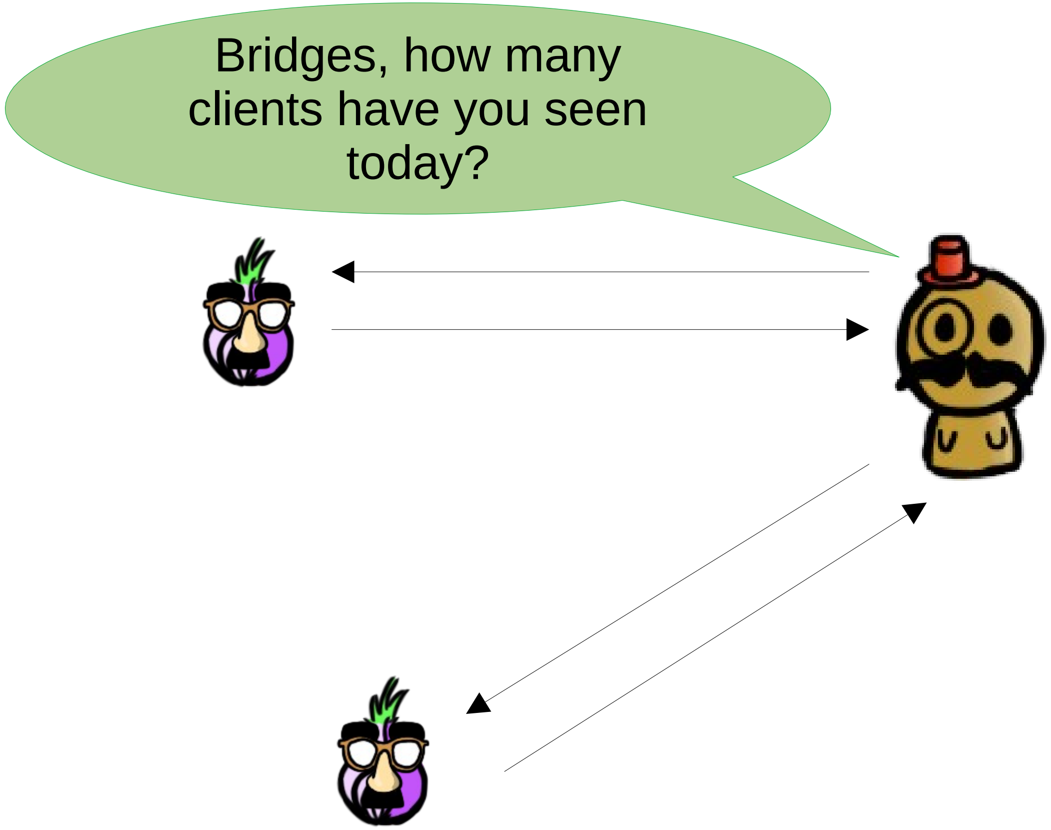


Bridge Usage Statistics

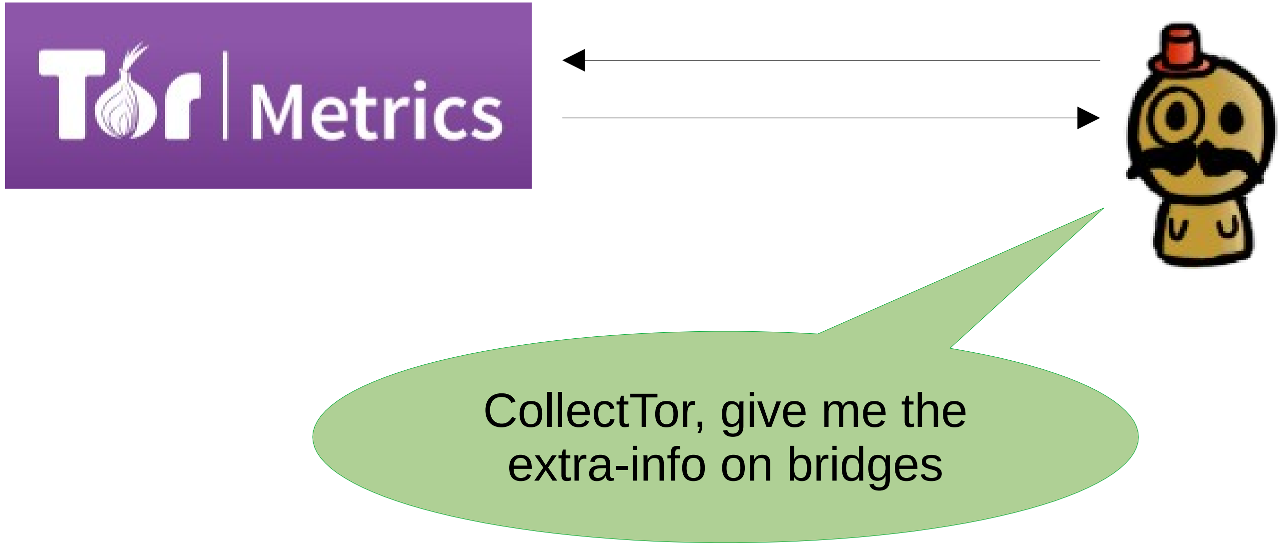
- Usage over a 24h period (86400 s)
- Bridges memorize unique IPs of clients
- Internal GeoIP database maps IPs to country codes (regions)

ru=24, us=8, cn=32, de=48

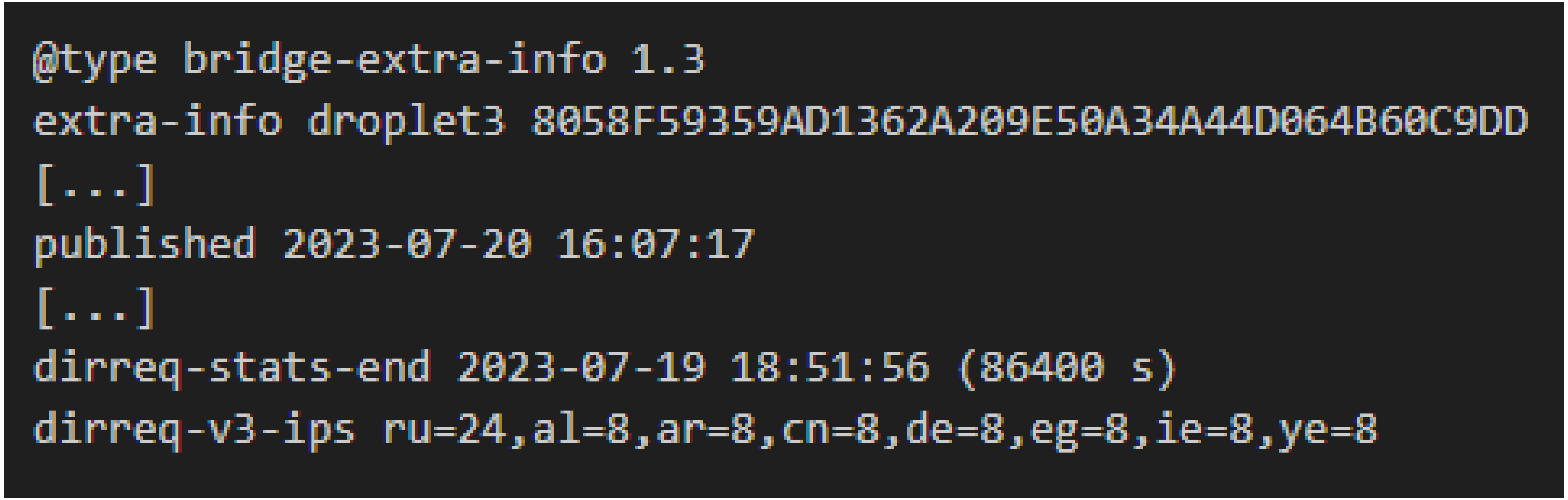
Bridge Usage Statistics



Bridge Usage Statistics

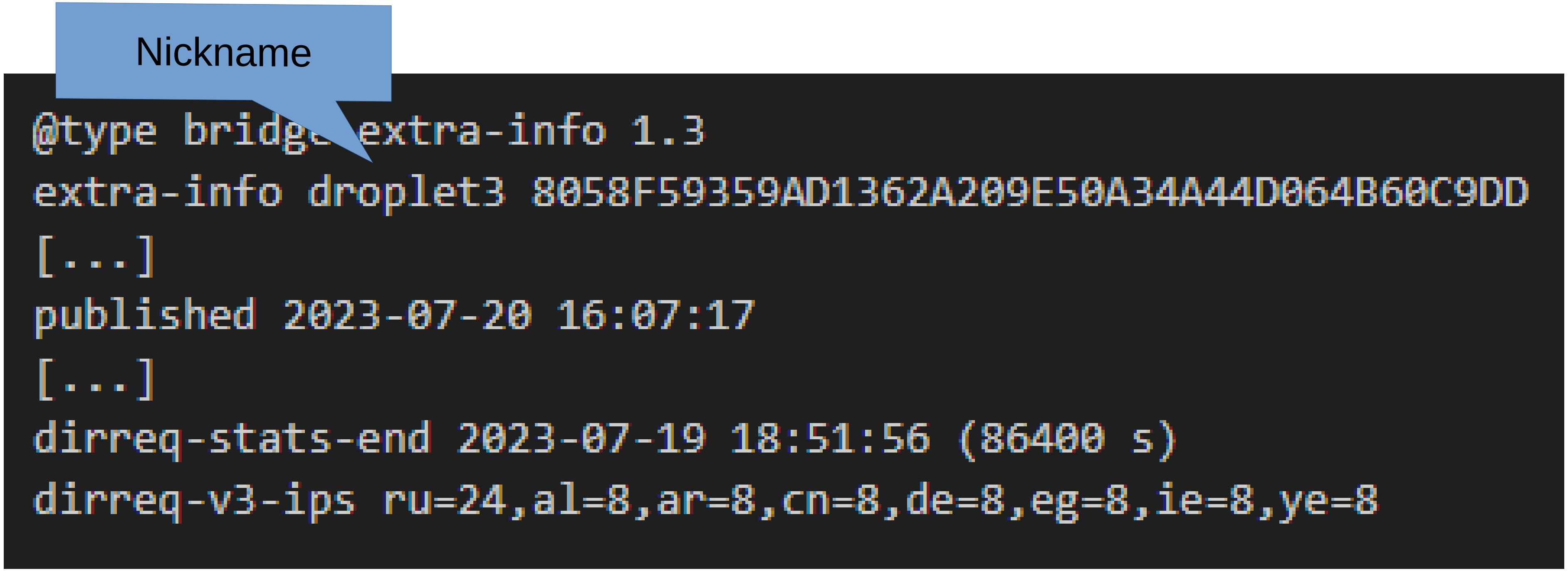


Bridge Usage Statistics



```
@type bridge-extra-info 1.3
extra-info droplet3 8058F59359AD1362A209E50A34A44D064B60C9DD
[...]
published 2023-07-20 16:07:17
[...]
dirreq-stats-end 2023-07-19 18:51:56 (86400 s)
dirreq-v3-ips ru=24,al=8,ar=8,cn=8,de=8,eg=8,ie=8,ye=8
```

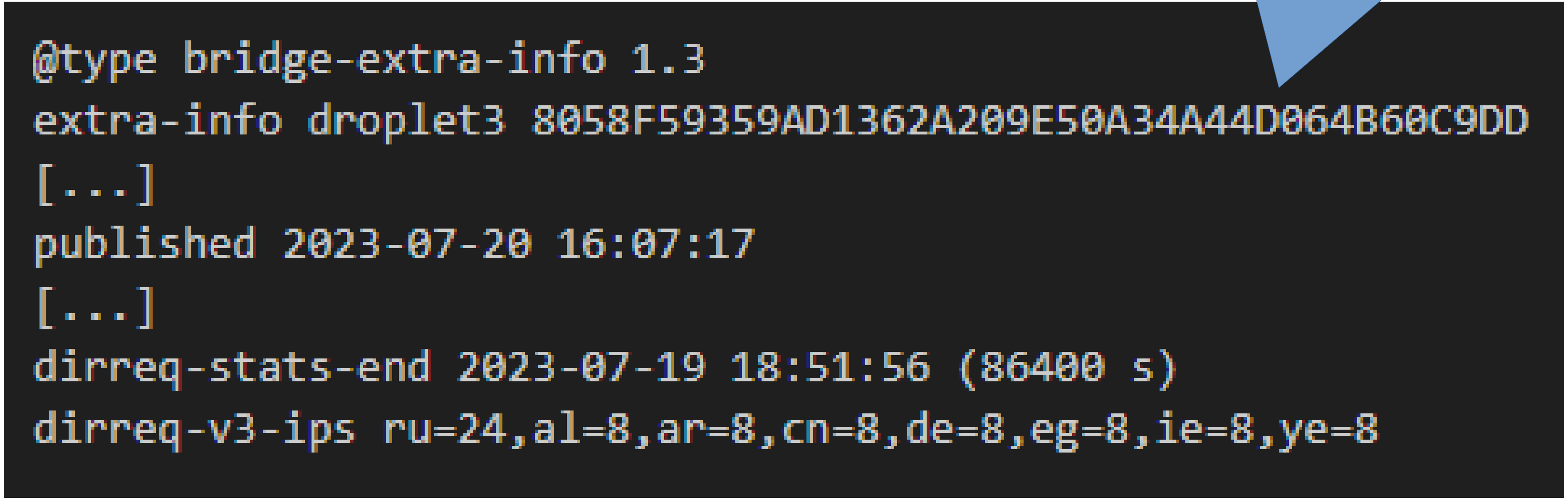
Bridge Usage Statistics



```
@type bridge-extra-info 1.3
extra-info droplet3 8058F59359AD1362A209E50A34A44D064B60C9DD
[...]
published 2023-07-20 16:07:17
[...]
dirreq-stats-end 2023-07-19 18:51:56 (86400 s)
dirreq-v3-ips ru=24,al=8,ar=8,cn=8,de=8,eg=8,ie=8,ye=8
```

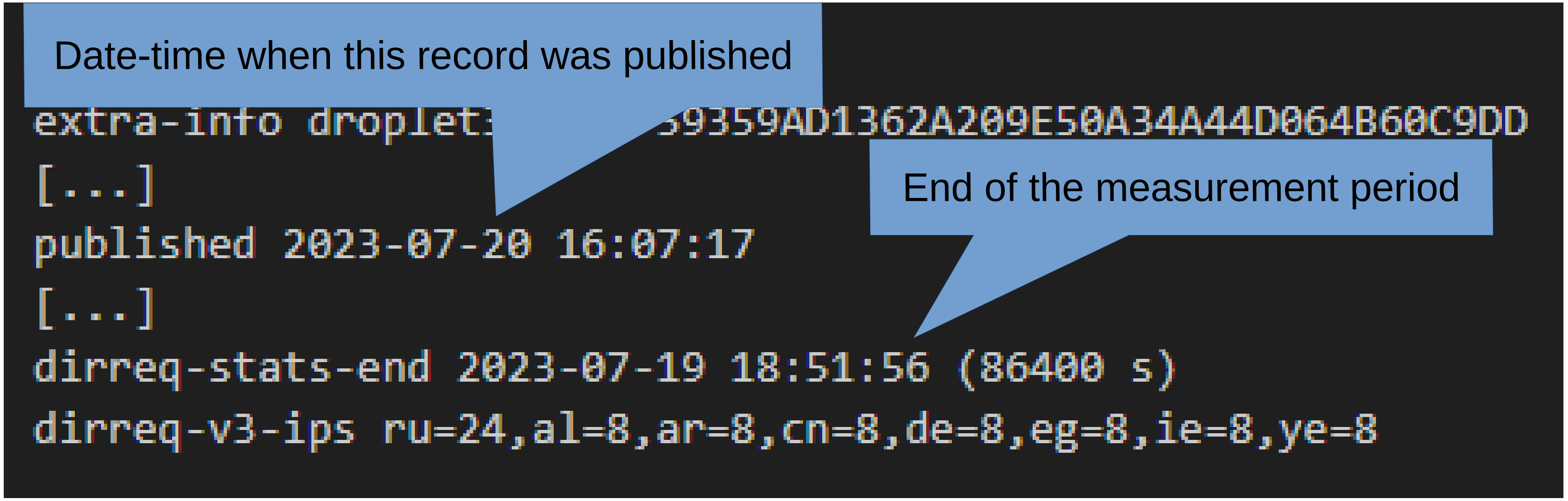
Bridge Usage Statistics

Bridge fingerprint (public key)



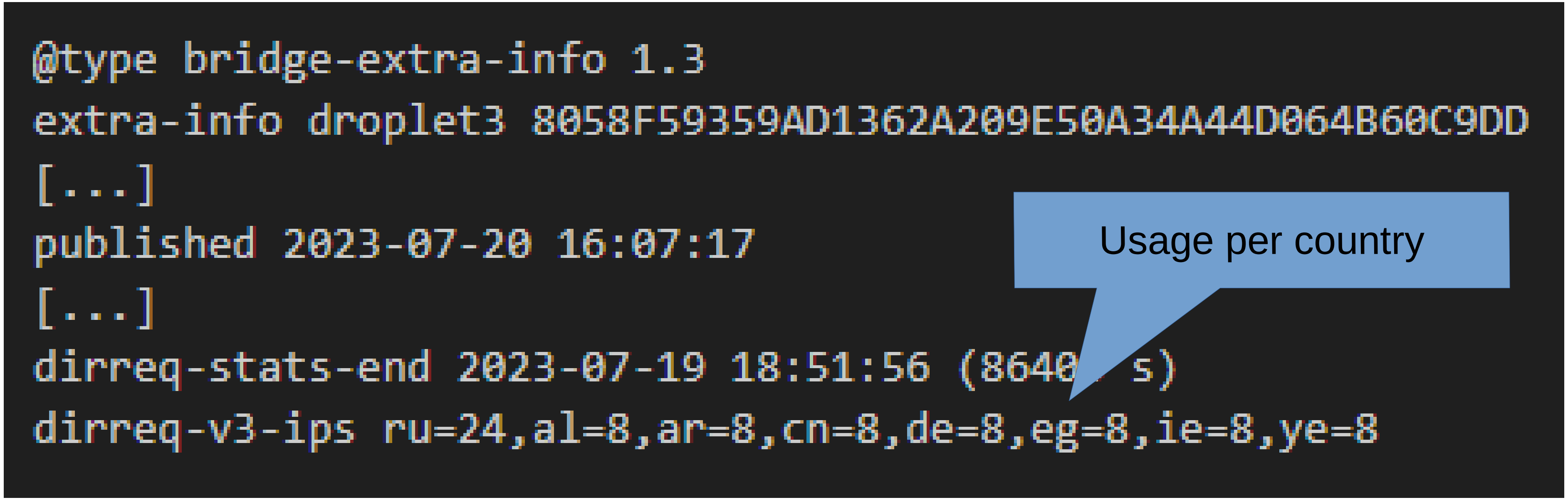
```
@type bridge-extra-info 1.3
extra-info droplet3 8058F59359AD1362A209E50A34A44D064B60C9DD
[...]
published 2023-07-20 16:07:17
[...]
dirreq-stats-end 2023-07-19 18:51:56 (86400 s)
dirreq-v3-ips ru=24,al=8,ar=8,cn=8,de=8,eg=8,ie=8,ye=8
```

Bridge Usage Statistics


Date-time when this record was published
End of the measurement period

```
extra-info droplet:          59359AD1362A209E50A34A44D064B60C9DD
[...]
published 2023-07-20 16:07:17
[...]
dirreq-stats-end 2023-07-19 18:51:56 (86400 s)
dirreq-v3-ips ru=24,al=8,ar=8,cn=8,de=8,eg=8,ie=8,ye=8
```

Bridge Usage Statistics



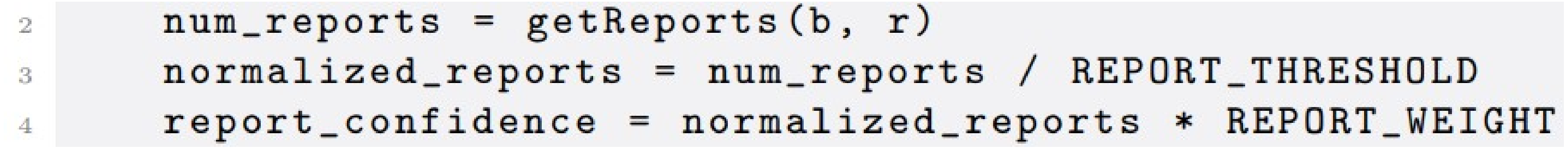
```
@type bridge-extra-info 1.3
extra-info droplet3 8058F59359AD1362A209E50A34A44D064B60C9DD
[...]
published 2023-07-20 16:07:17
[...]
dirreq-stats-end 2023-07-19 18:51:56 (8640  s)
dirreq-v3-ips ru=24,al=8,ar=8,cn=8,de=8,eg=8,ie=8,ye=8
```

Phase 1 - Algorithm

```
def Phase1(Bridge b, Region r):
    num_reports = getReports(b, r)
    normalized_reports = num_reports / REPORT_THRESHOLD
    report_confidence = normalized_reports * REPORT_WEIGHT

    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT

    if avg_users > MIN_USAGE_THRESHOLD:
        if curr_users < MIN_USAGE_THRESHOLD:
            bstats_confidence = 1 * BSTATS_WEIGHT

    confidence = report_confidence + bstats_confidence

    return confidence
```

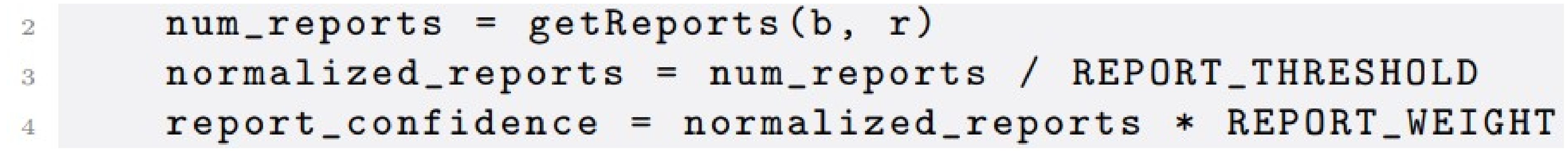
Algorithm 1.1: Pseudocode for phase 1.

Phase 1 – User Reports



```
    num_reports = getReports(b, r)
    normalized_reports = num_reports / REPORT_THRESHOLD
    report_confidence = normalized_reports * REPORT_WEIGHT
```

Phase 1 – User Reports

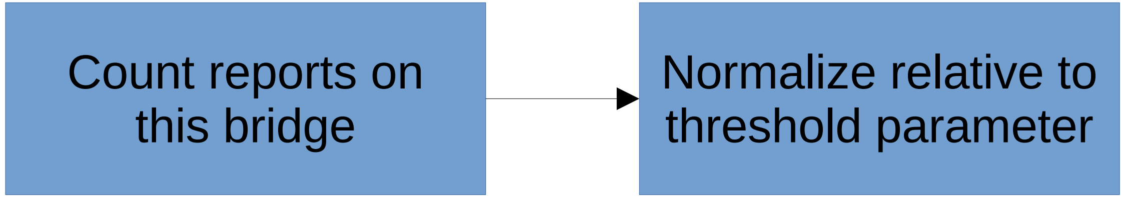


```
    num_reports = getReports(b, r)
    normalized_reports = num_reports / REPORT_THRESHOLD
    report_confidence = normalized_reports * REPORT_WEIGHT
```

Count reports on this bridge

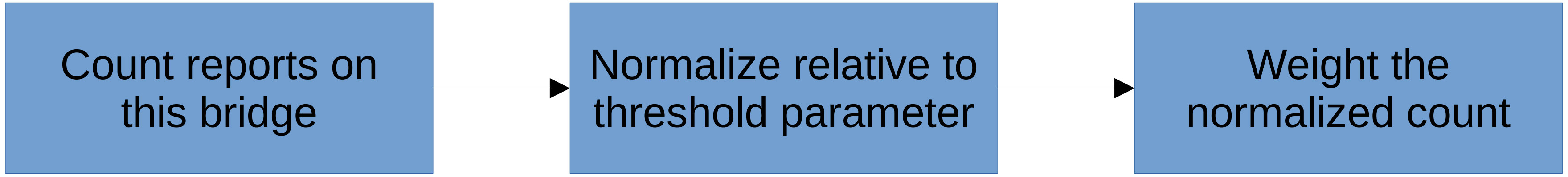
Phase 1 – User Reports

```
    num_reports = getReports(b, r)
    normalized_reports = num_reports / REPORT_THRESHOLD
    report_confidence = normalized_reports * REPORT_WEIGHT
```

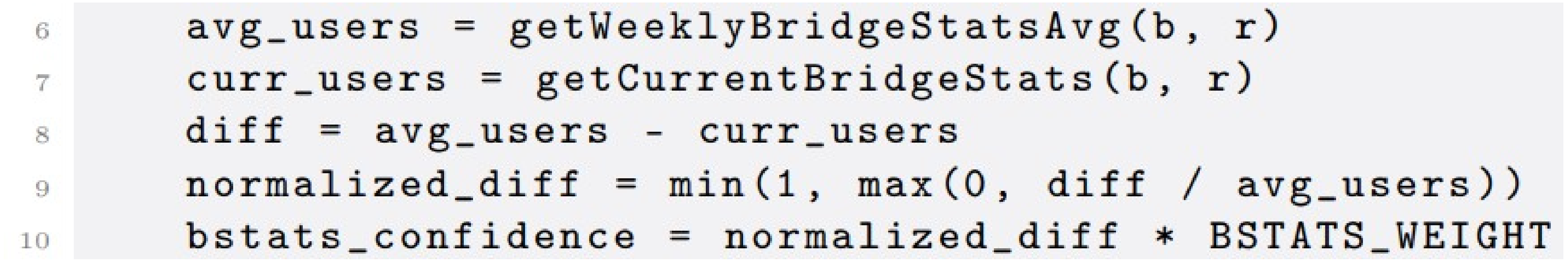

Count reports on this bridge
Normalize relative to threshold parameter

Phase 1 – User Reports

```
    num_reports = getReports(b, r)
    normalized_reports = num_reports / REPORT_THRESHOLD
    report_confidence = normalized_reports * REPORT_WEIGHT
```

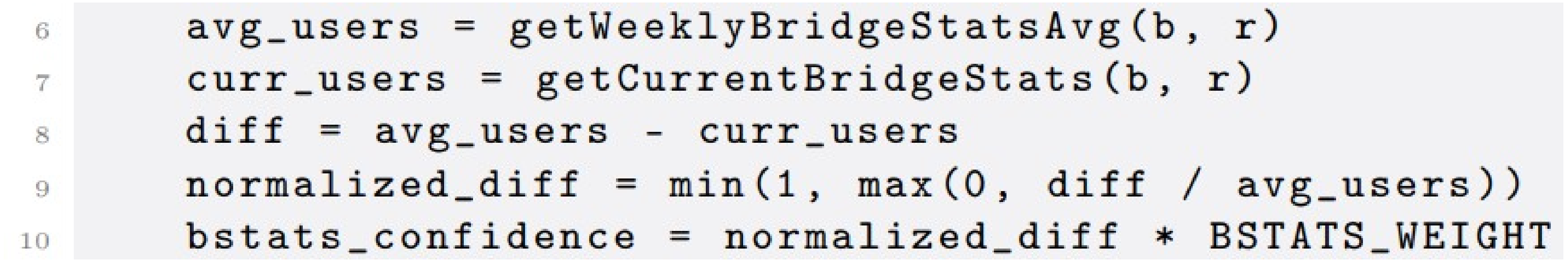

Count reports on this bridge
Normalize relative to threshold parameter
Weight the normalized count

Phase 1 – Bridge Stats



```
    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT
```

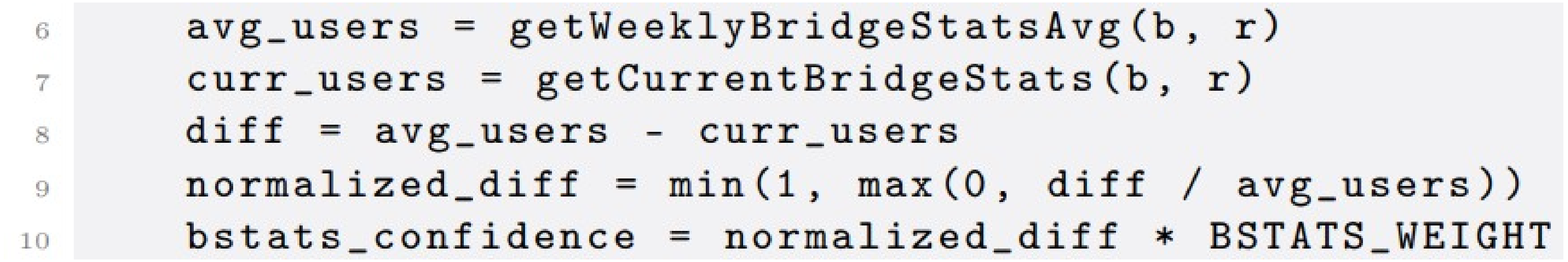
Phase 1 – Bridge Stats



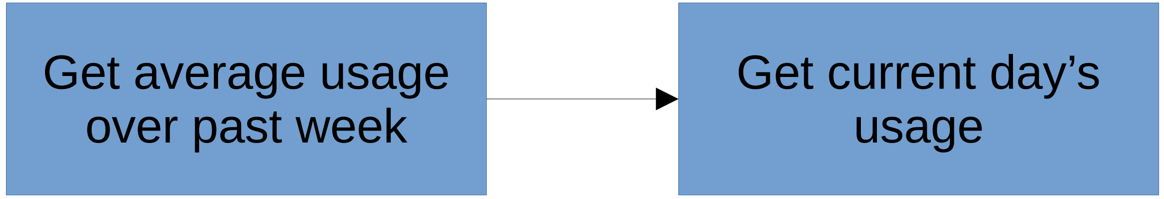
```
    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT
```

Get average usage over past week

Phase 1 – Bridge Stats

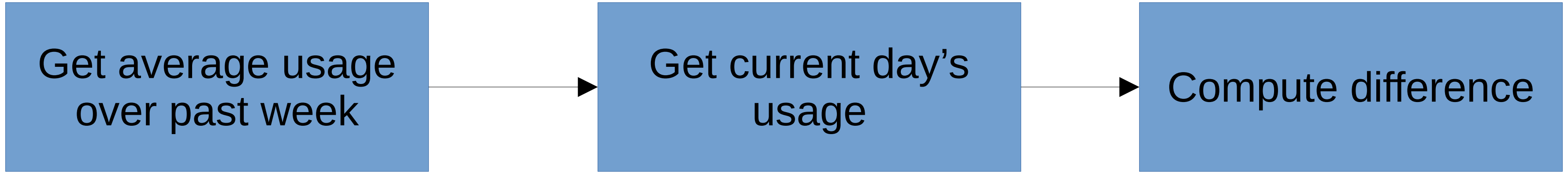


```
    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT
```



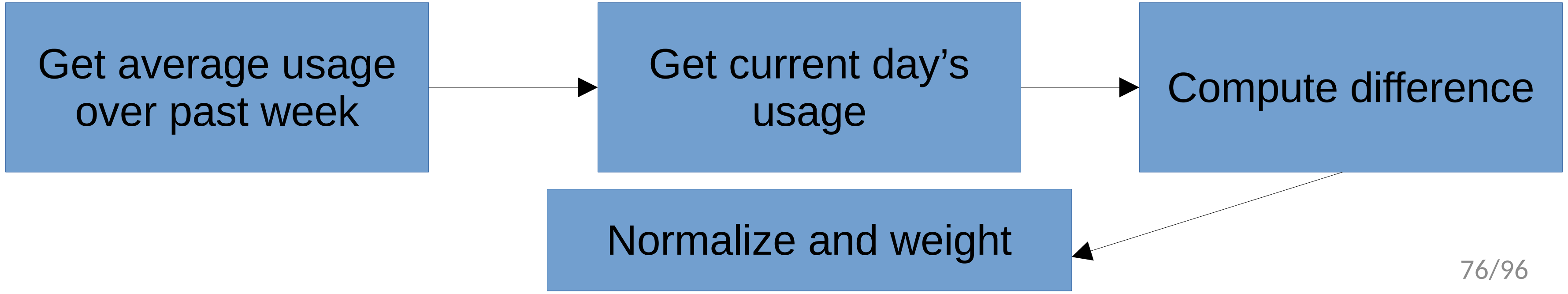
Phase 1 – Bridge Stats

```
    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT
```

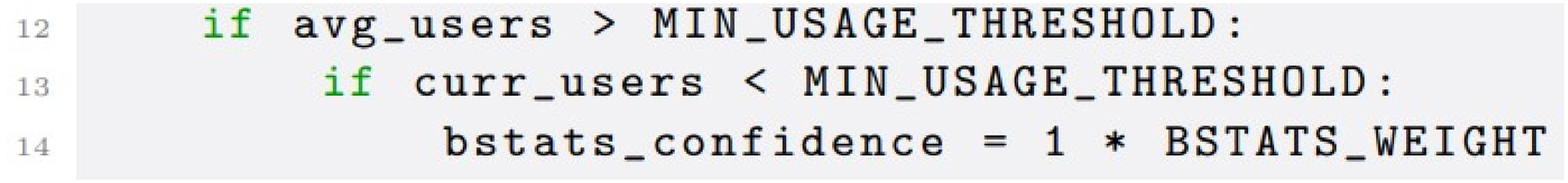


Phase 1 – Bridge Stats

```
    avg_users = getWeeklyBridgeStatsAvg(b, r)
    curr_users = getCurrentBridgeStats(b, r)
    diff = avg_users - curr_users
    normalized_diff = min(1, max(0, diff / avg_users))
    bstats_confidence = normalized_diff * BSTATS_WEIGHT
```



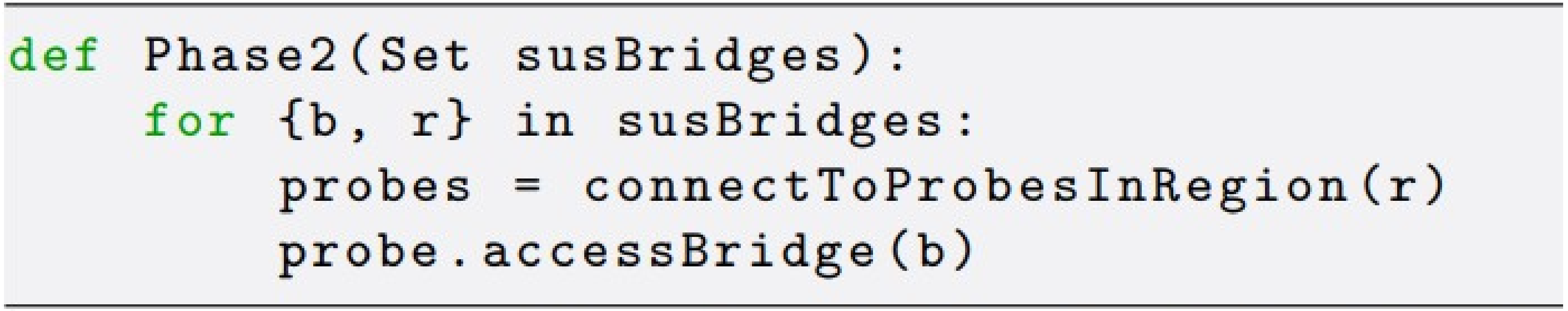
Phase 1 – Bridge Stats



```
    if avg_users > MIN_USAGE_THRESHOLD:
        if curr_users < MIN_USAGE_THRESHOLD:
            bstats_confidence = 1 * BSTATS_WEIGHT
```

Threshold parameter for minimum bridge usage

Phase 2 - Probing Bridges

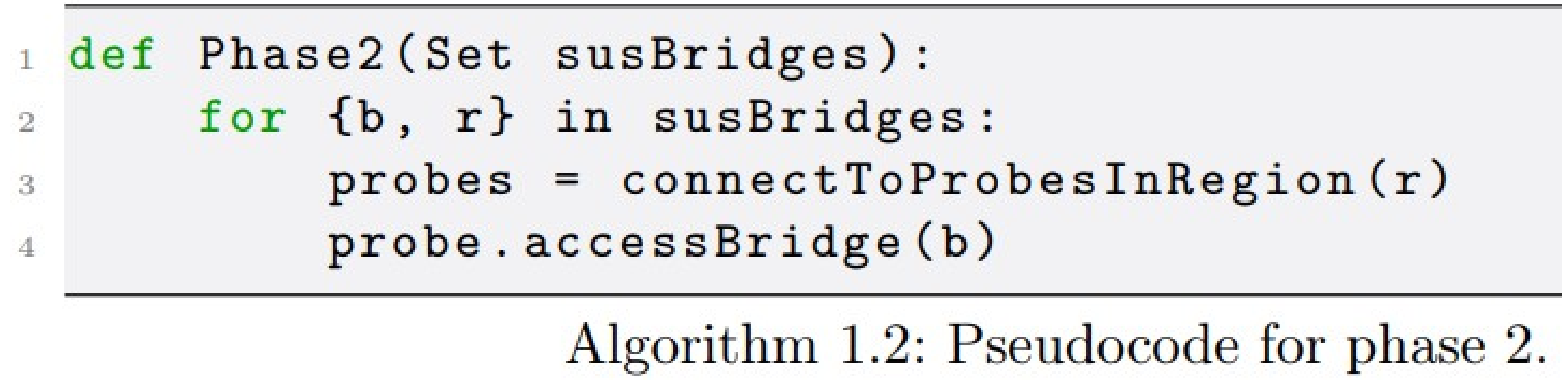


```
def Phase2(Set susBridges):
    for {b, r} in susBridges:
        probes = connectToProbesInRegion(r)
        probe.accessBridge(b)
```

Algorithm 1.2: Pseudocode for phase 2.

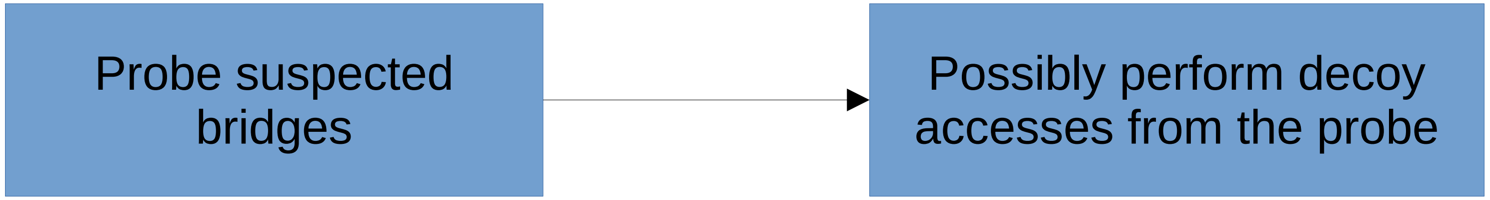
Probe suspected bridges

Phase 2 - Probing Bridges

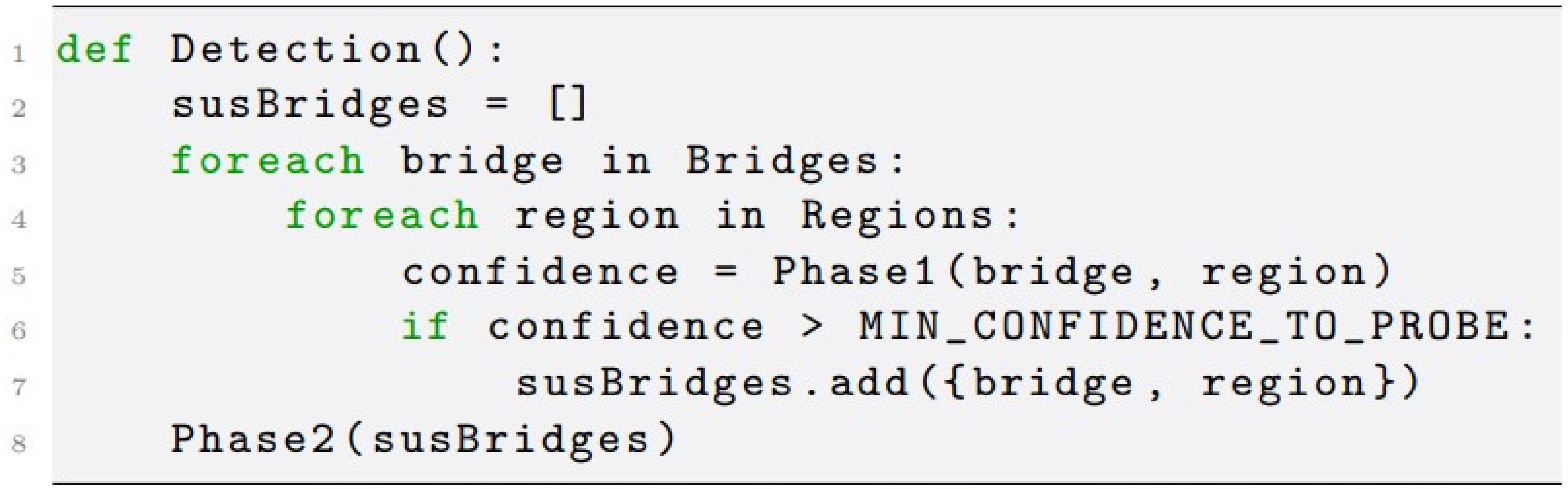


```
def Phase2(Set susBridges):
    for {b, r} in susBridges:
        probes = connectToProbesInRegion(r)
        probe.accessBridge(b)
```

Algorithm 1.2: Pseudocode for phase 2.



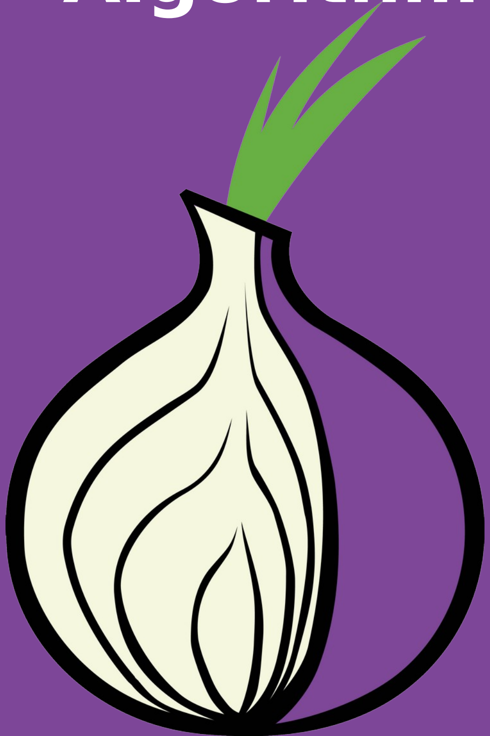
Blockage Detection Algorithm



```
def Detection():
    susBridges = []
    foreach bridge in Bridges:
        foreach region in Regions:
            confidence = Phase1(bridge, region)
            if confidence > MIN_CONFIDENCE_TO_PROBE:
                susBridges.add({bridge, region})
    Phase2(susBridges)
```

Algorithm 1.3: Pseudocode for our detection algorithm.

Simulating Detection Algorithm



Simulation

- Deploying a detection algorithm in practice has logistical issues
  - Need to know about all bridges
  - Need to probe from inside censored regions
- Simulate the aspects of the system that we care about

Simulation – The blocked bridge game

- Simulate user, censor and detector interaction with bridges
- No networking simulation

Simulation – The blocked bridge game

Users access bridges

Simulation – The blocked bridge game

Users access bridges

Bridges aggregate usage stats

Simulation – The blocked bridge game

Users access bridges

Censor blocks some bridges with low probability

Bridges aggregate usage stats

Simulation – The blocked bridge game

Users access bridges

Censor blocks some bridges with low probability

Bridges aggregate usage stats

Users report bridges

Simulation – The blocked bridge game

Users access bridges

Censor blocks some bridges with low probability

Detector checks bridges

Bridges aggregate usage stats

Users report bridges

Simulation – The blocked bridge game

Users access bridges

Censor blocks some bridges with low probability

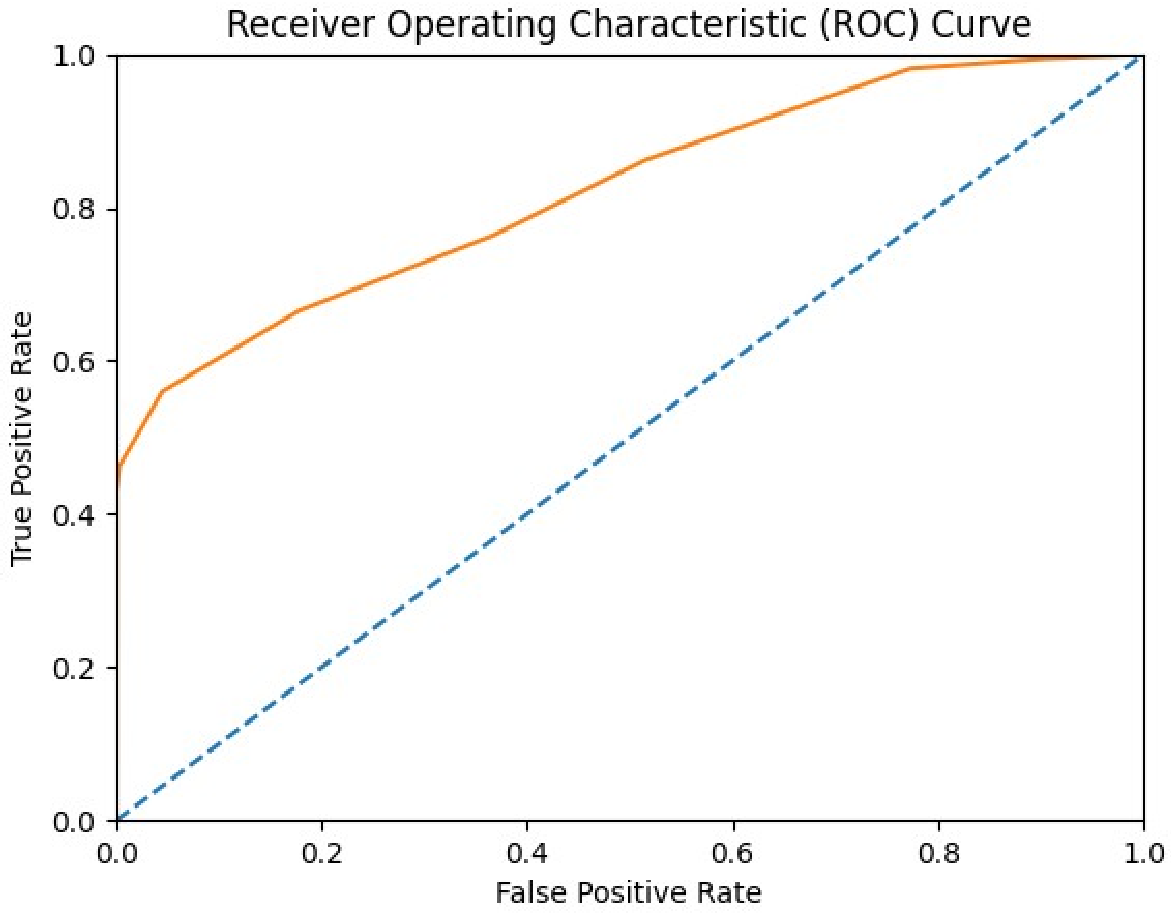
Detector checks bridges

Bridges aggregate usage stats

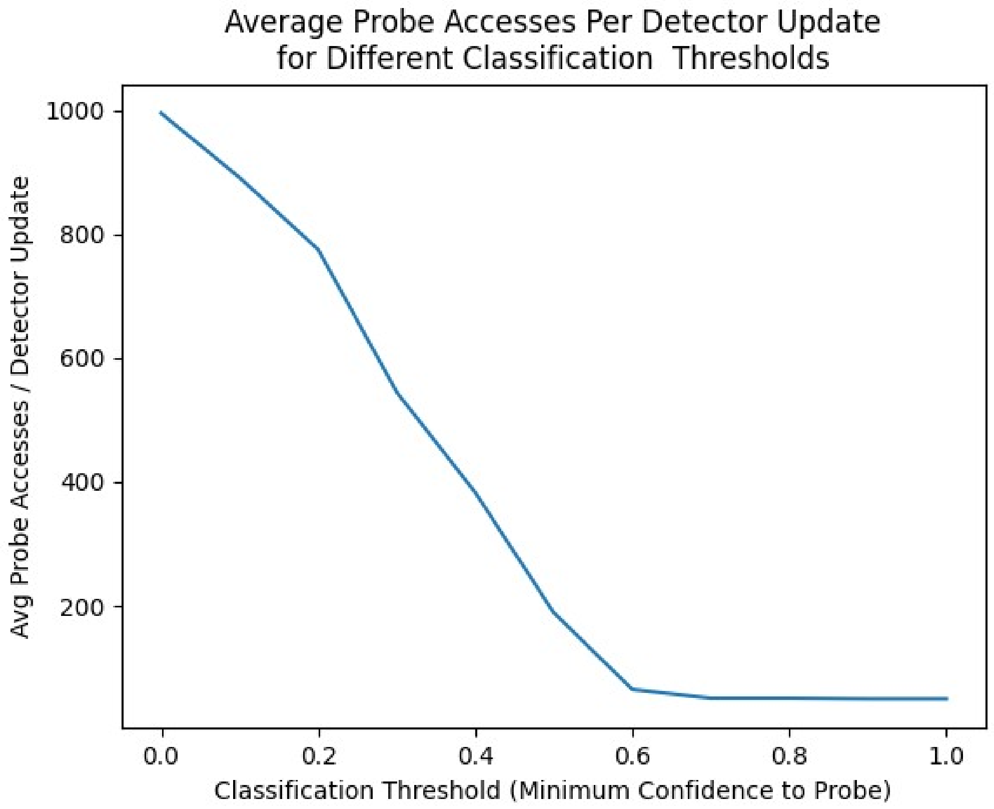
Users report bridges

Detector reports blocked bridges to bridge authority

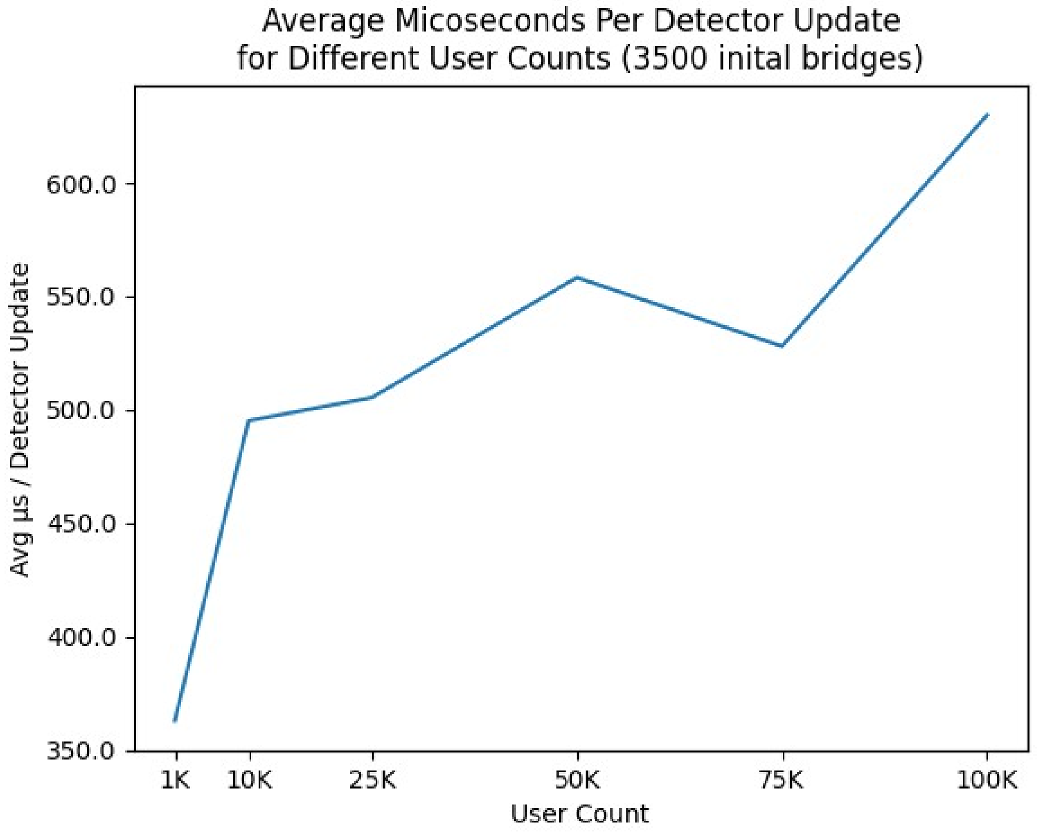
ROC


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Probes launched



Cost of analysis



Limitations & Future Work

Limitations

- User reports could be more useful if we could determine if the reports are from unique individuals
- Iterating over all bridges is feasible only so long as Tor's bridge count remains low
- Censor could block all bridges at once (ex. if protocol is fingerprinted and blocked)
- Probing via VPS might not be possible
    - Volunteer probing risks repercussions

Future Work

- Integrate with bridge authority like Lox
  - Consider trust level when weighting user reports
- Consider using reverse scans from bridges
- Track when bridges are detected as blocked
  - If too many are blocked in close time period report a special case

Conclusion

- Blocked bridge detection algorithm
- Simulation of bridge accesses in censored regions
- Results based on simulation