

Deanonymizing Tor

Nathan S. Evans

Christian Grothoff

`Nathan.S.Evans@du.edu`

`christian@grothoff.org`

Colorado Research Institute for Security and Privacy

University of Denver



Motivation

- Tor is probably the most popular and widely used free software P2P network used to achieve anonymity on the Internet:
 - Tor has a strong, large user base
 - The project is well supported
 - Generally assumed to give users strong anonymity

The news today:

All the Tor nodes involved in a circuit can be discovered, reducing Tor users level of anonymity

Tor General Information

- Tor stands for “The onion router”
 - Encrypts data multiple times and is decrypted as it travels through the network a layer at a time: like peeling an onion
- Tor is a low latency P2P mix cascade
- Routes data through network along a “circuit”
- Data is encrypted as it passes through nodes (until the last hop)

Routing

- Data is forwarded through the network
- Each node knows only the previous hop and the next hop
- Only the originator knows all the hops
- Number of hops is hard coded (currently set to three)

Key security goal: No node in the path can discover the full path

Routing Example

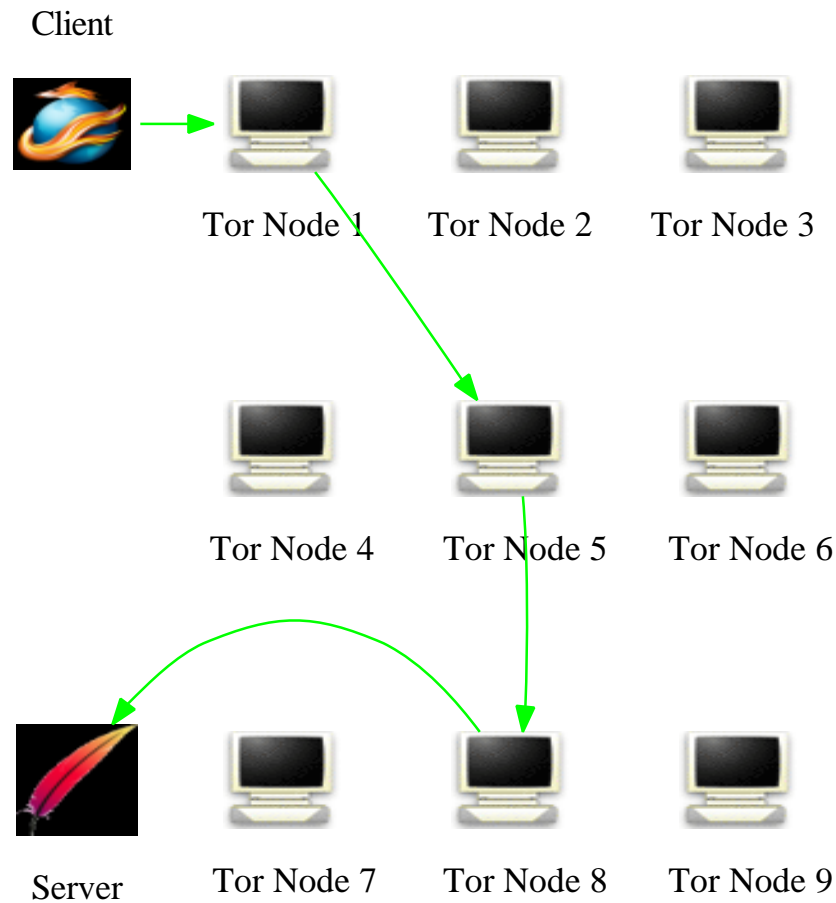


Figure 1: Example showing how a normal path is chosen in the Tor network

Previous work

- Murdoch and Danezis wrote “Low Cost Traffic Analysis of Tor”
- Goal is to discover all the Tor routers involved in a given circuit
- Based on being able to tell the added load of one normal Tor connection
- Send a certain sequence down a tunnel, monitor each Tor router to see if it is involved
- Their attack worked well with the 2005 Tor network consisting of approximately a dozen Tor routers

Problems With Previous Work

- Less feasible with 1000+ routers
- Must identify all the separate routers in the circuit
- Attempting to measure small effects, large fluctuations that occur in actual current network give false positives
- We replicated their experiments, found method to be much less effective on today's network

Our Basis for Deanonymization

- Three design issues enable users to be deanonymized
 1. No artificial delays induced on connections
 2. Path length is set at a small finite number
 3. Paths of arbitrary length through the network can be constructed
- Target user is running Tor with privoxy with all the default settings

Regular Path Example

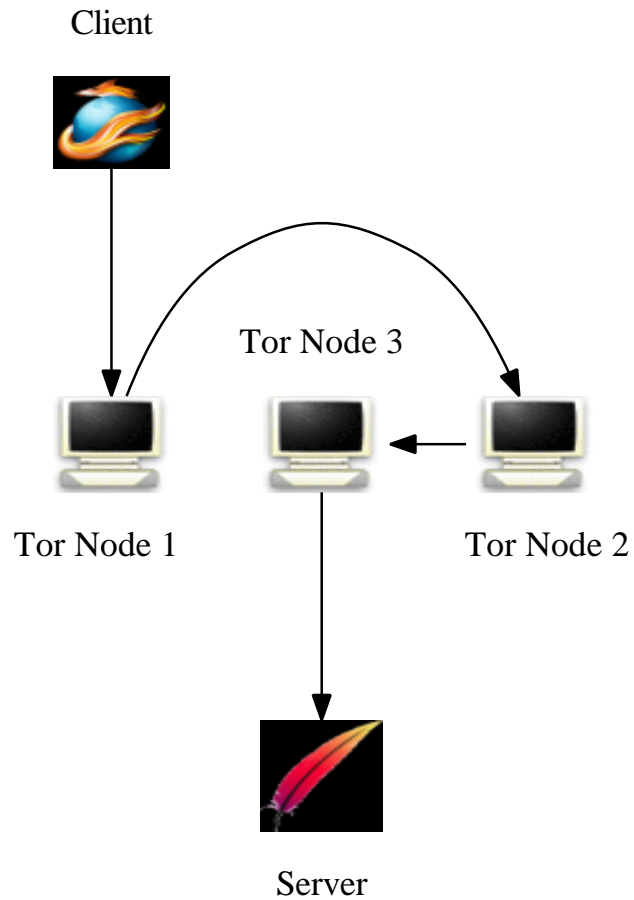


Figure 2: Example showing a typical Tor circuit

Circular Path Example 1/5

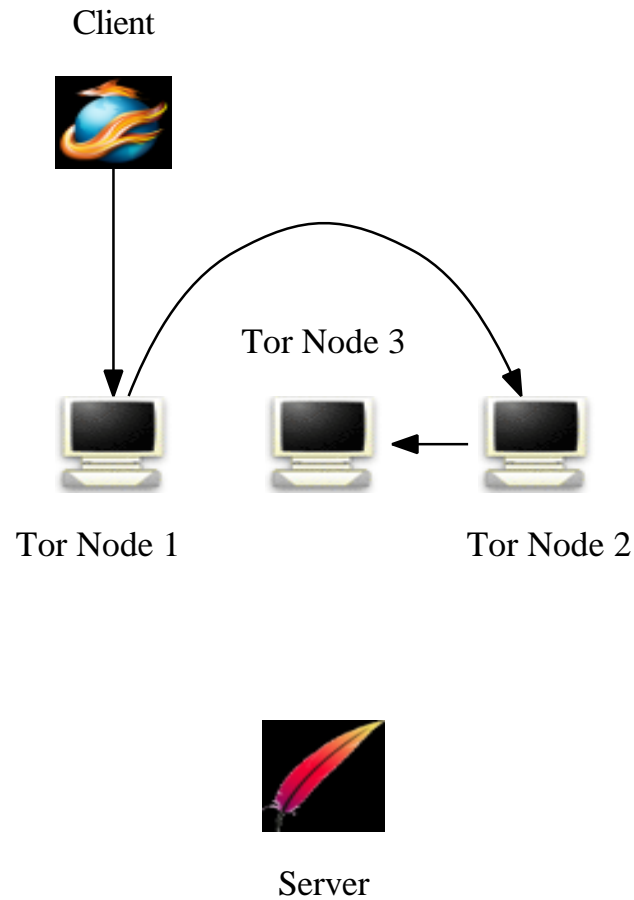


Figure 3: Example showing how long circular routes are created

Circular Path Example 2/5

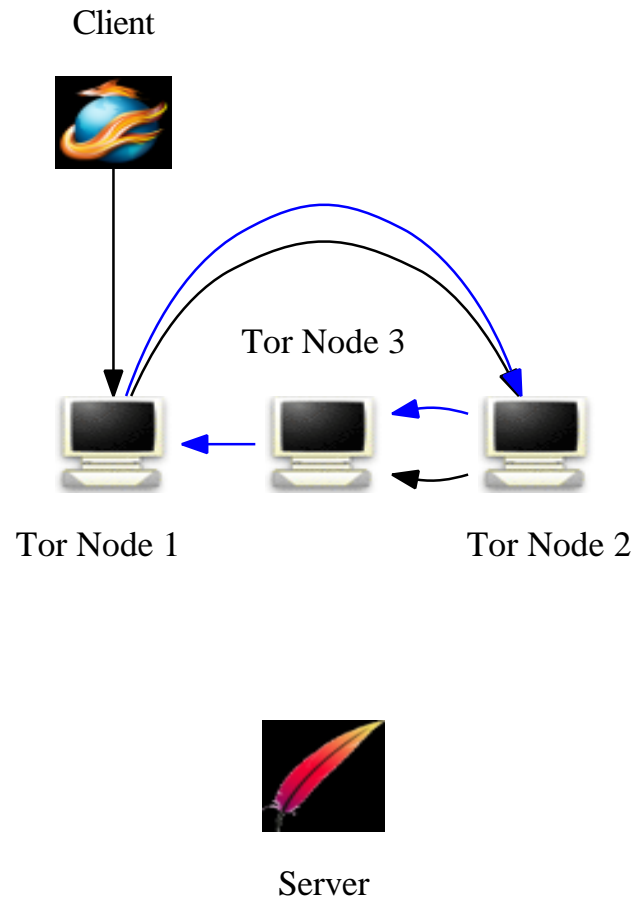


Figure 4: Example showing how long circular routes are created

Circular Path Example 3/5

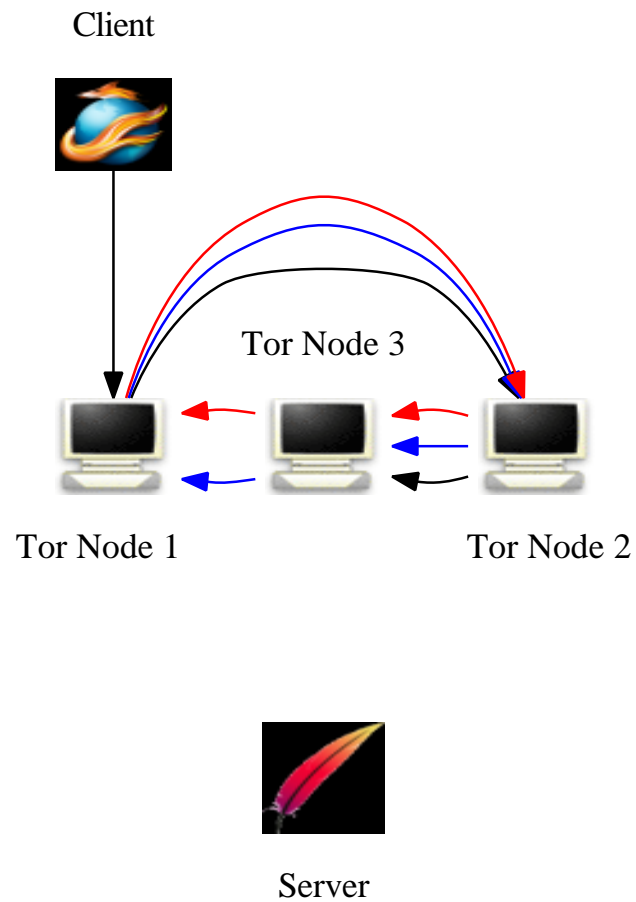


Figure 5: Example showing how long circular routes are created

Circular Path Example 4/5

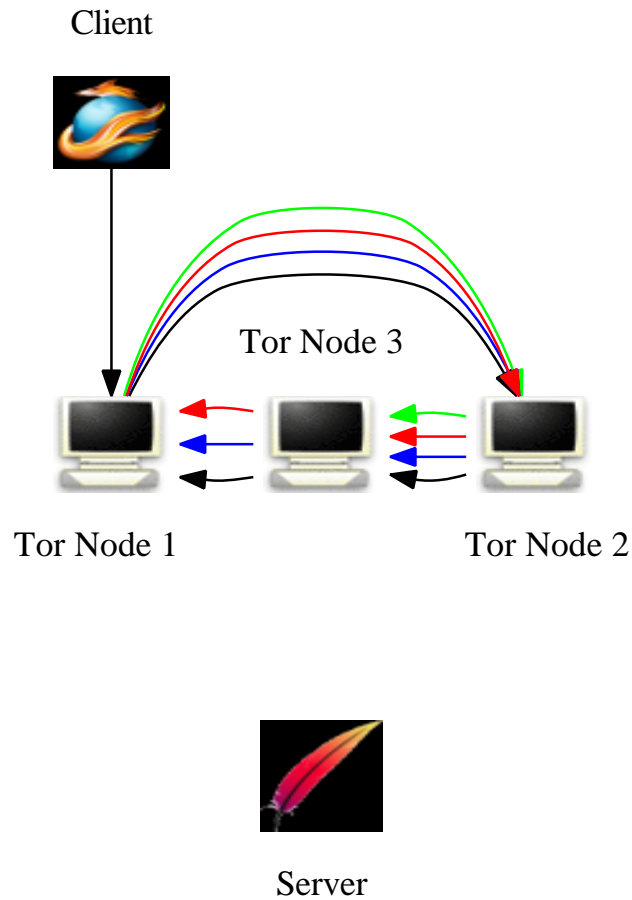


Figure 6: Example showing how long circular routes are created

Circular Path Example 5/5

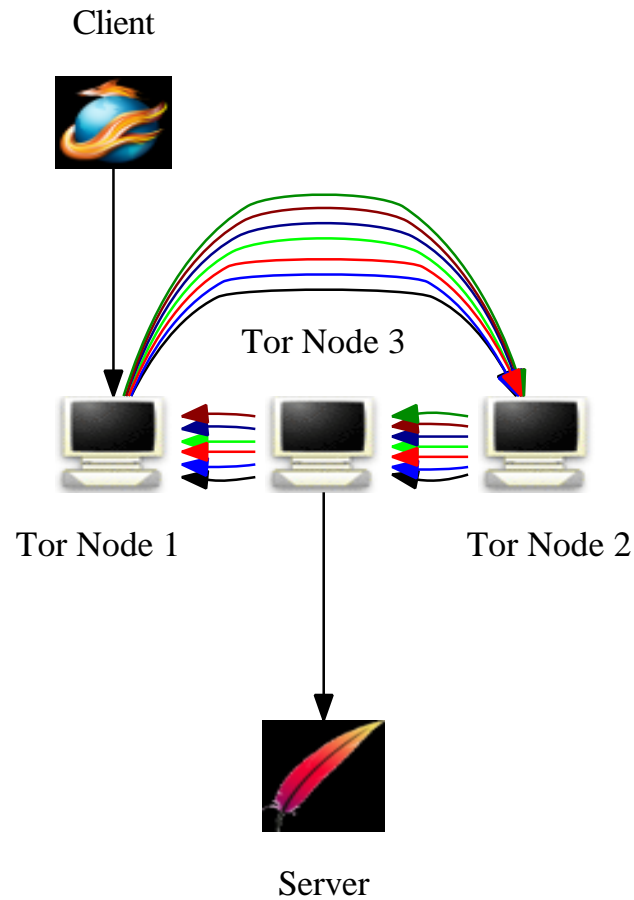


Figure 7: Example showing how long circular routes are created

Attack Example

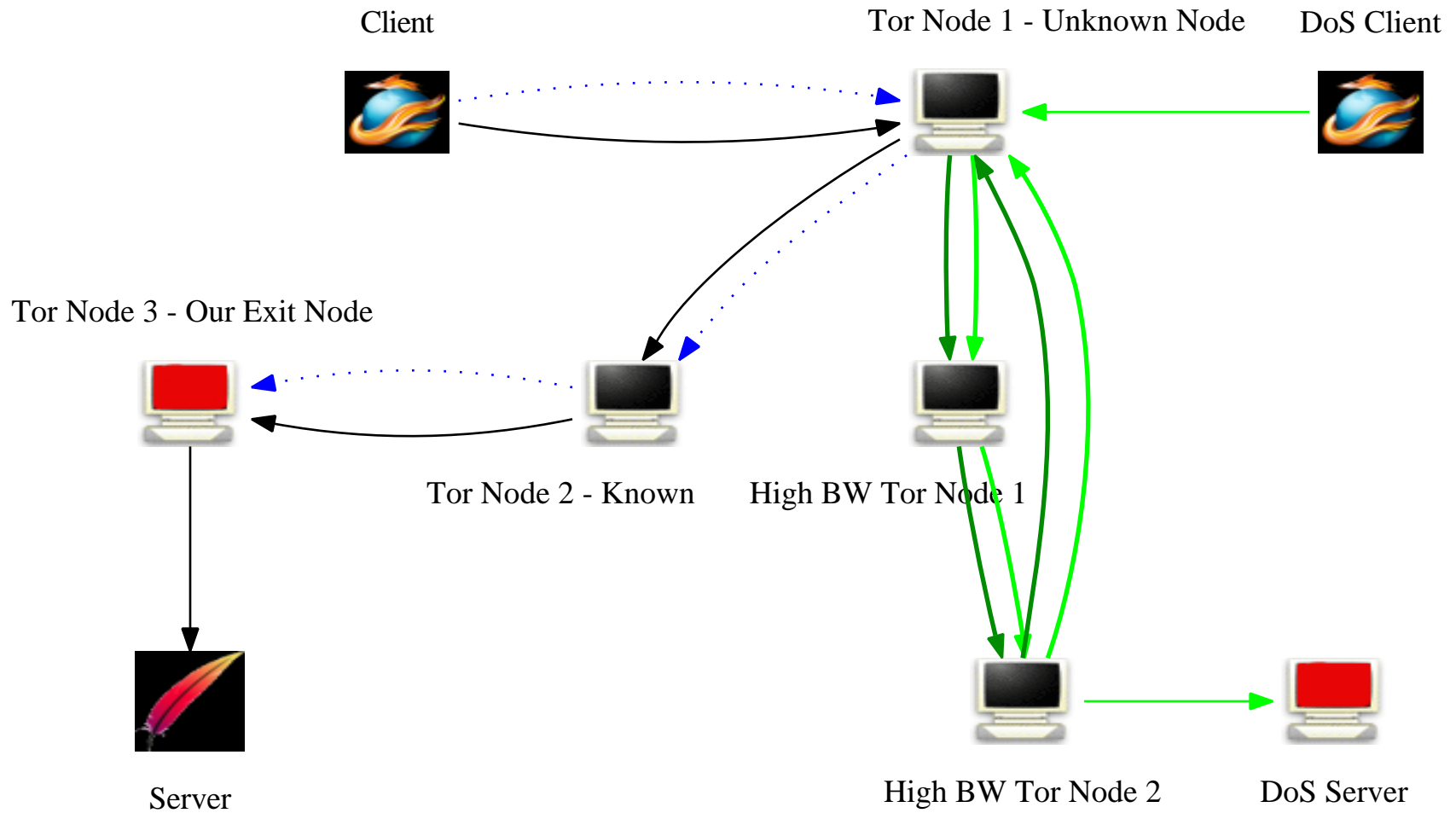


Figure 8: Example showing a partial view of the Tor network during the attack

Attack Implementation

- Modified exit node
- Modified DoS node
- Lightweight DoS web server running on GNU libmicrohttpd
- Client side JavaScript for latency measurements
- Instrumentation client to receive data

Attack Implementation

- Exit node injects JavaScript “ping” code into HTML response
- Client browses as usual, while JavaScript continues to “phone home”
- Exit node measures variance in latency
- While continuing to measure, DoS attack strains possible first hop(s)
- If no significant variance observed, pick another node from candidates and start over
- Once sufficient change is observed in measurements, initial node has been found



Gathered Data Example (1/10)

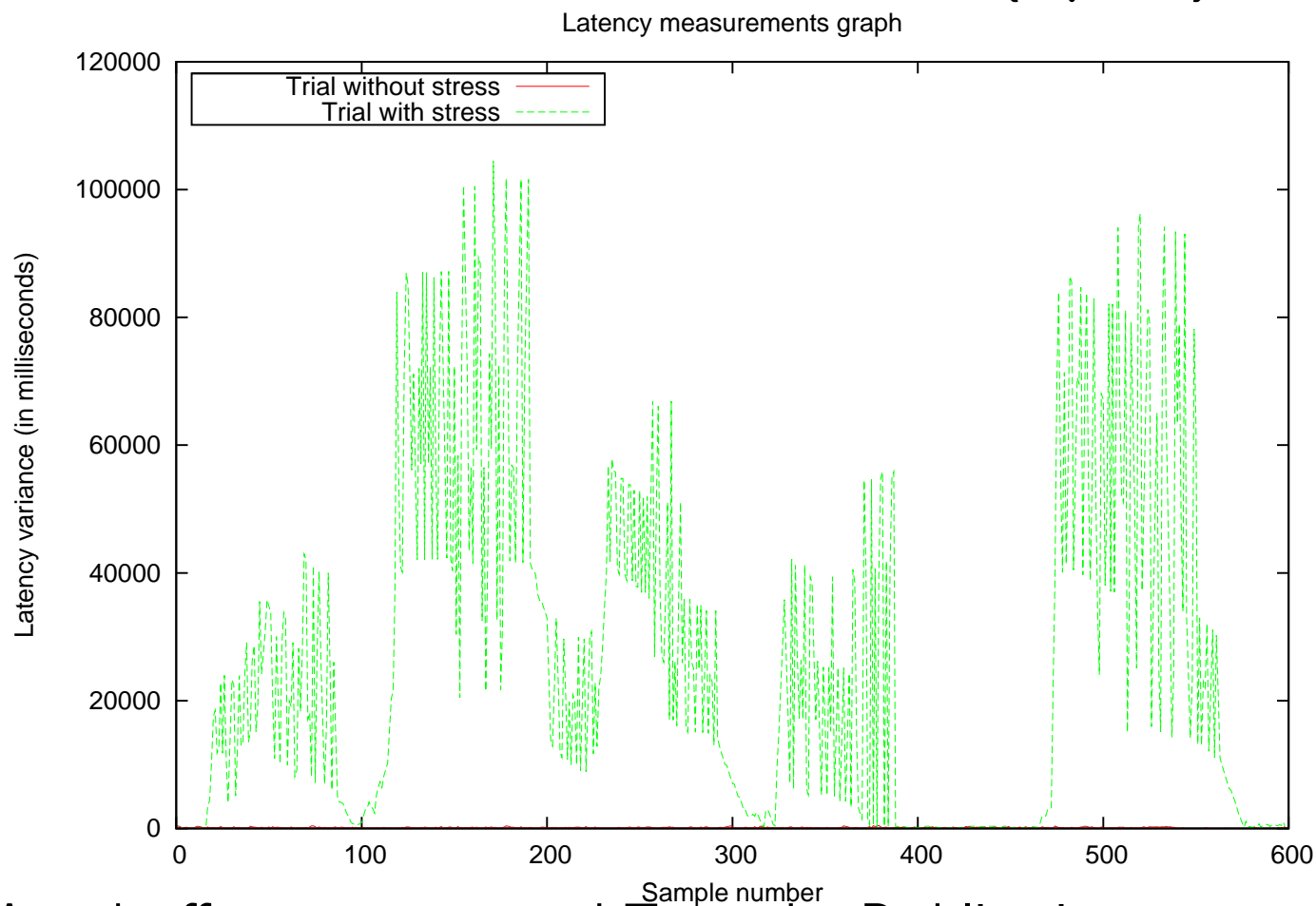


Figure 9: Attack effects on an unused Tor node. Red line is unstressed test, green is stressed test. X axis is measurement number, Y is “latency”

Gathered Data Example (2/10)

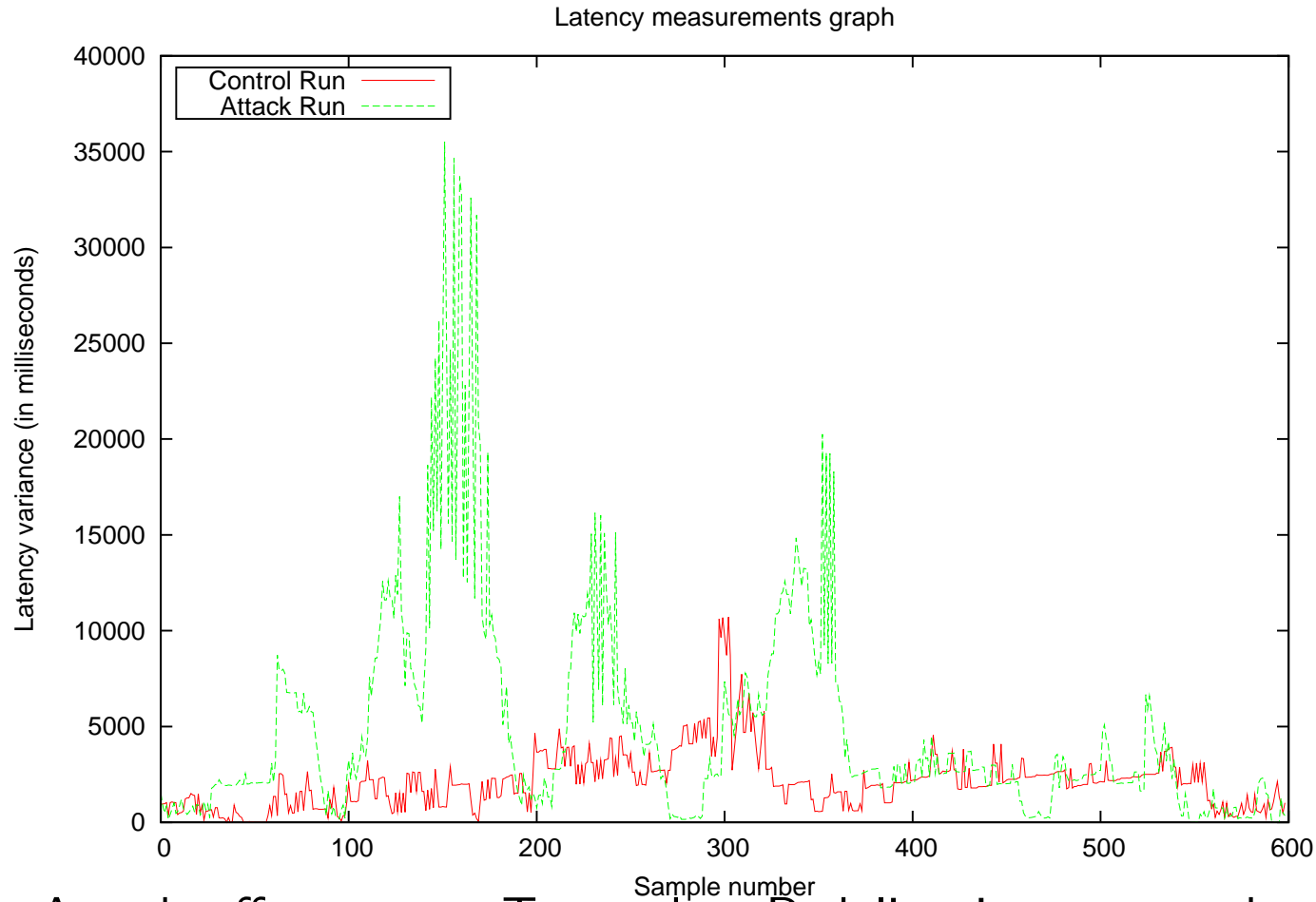


Figure 10: Attack effects on a Tor node. Red line is unstressed test, green is stressed test

Gathered Data Example (3/10)

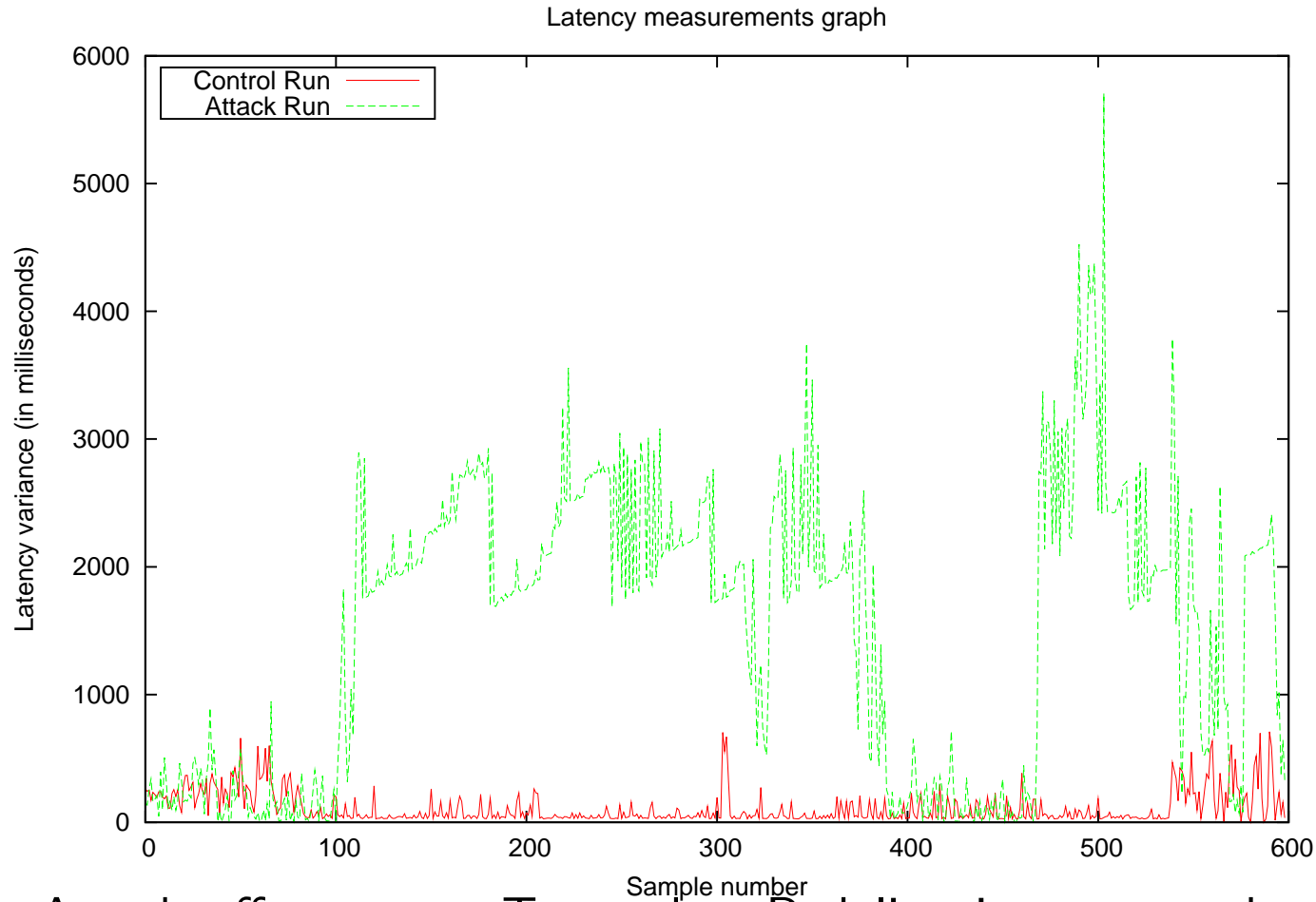


Figure 11: Attack effects on a Tor node. Red line is unstressed test, green is stressed test

Gathered Data Example (4/10)

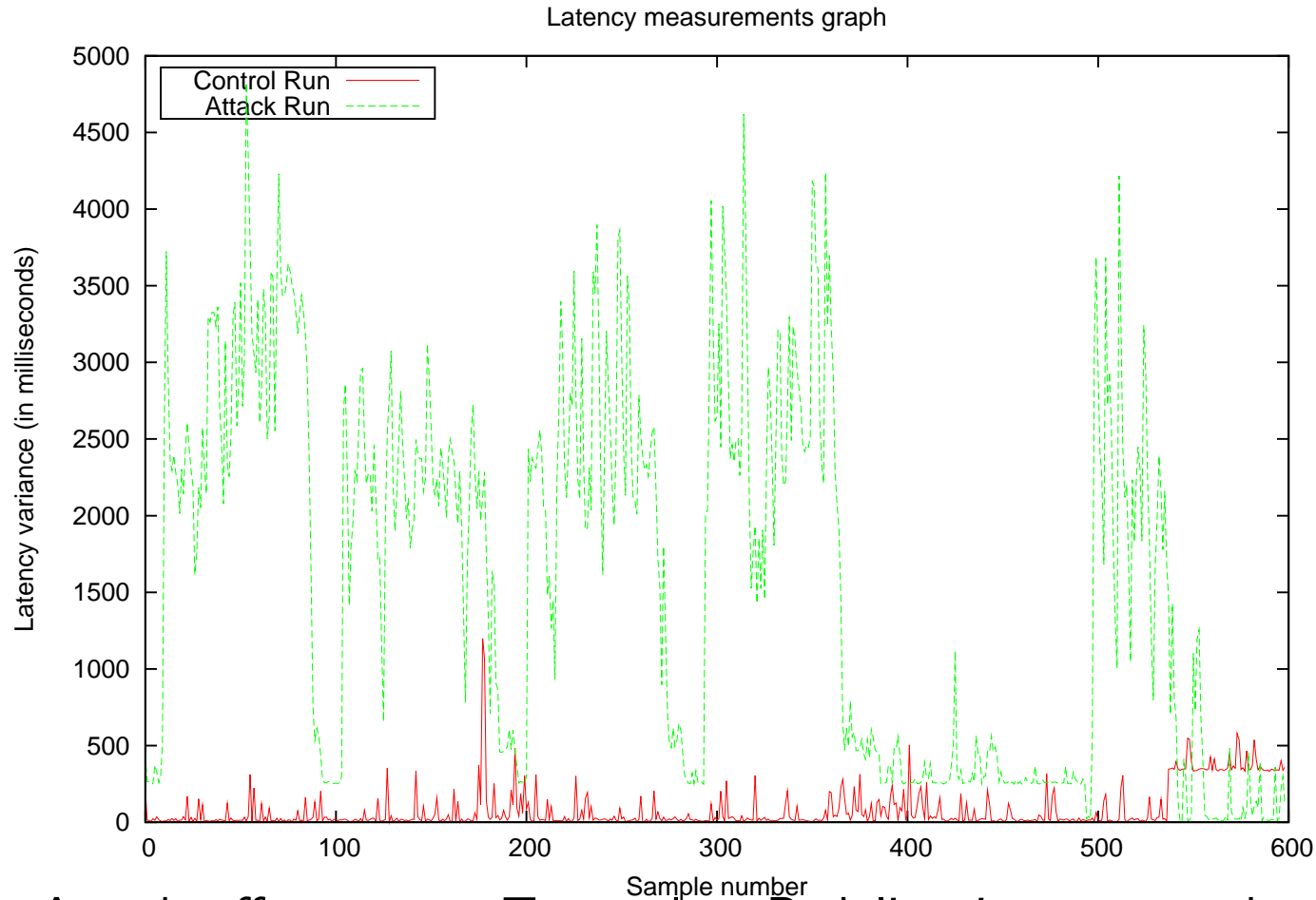


Figure 12: Attack effects on a Tor node. Red line is unstressed test, green is stressed test

Gathered Data Example (5/10)

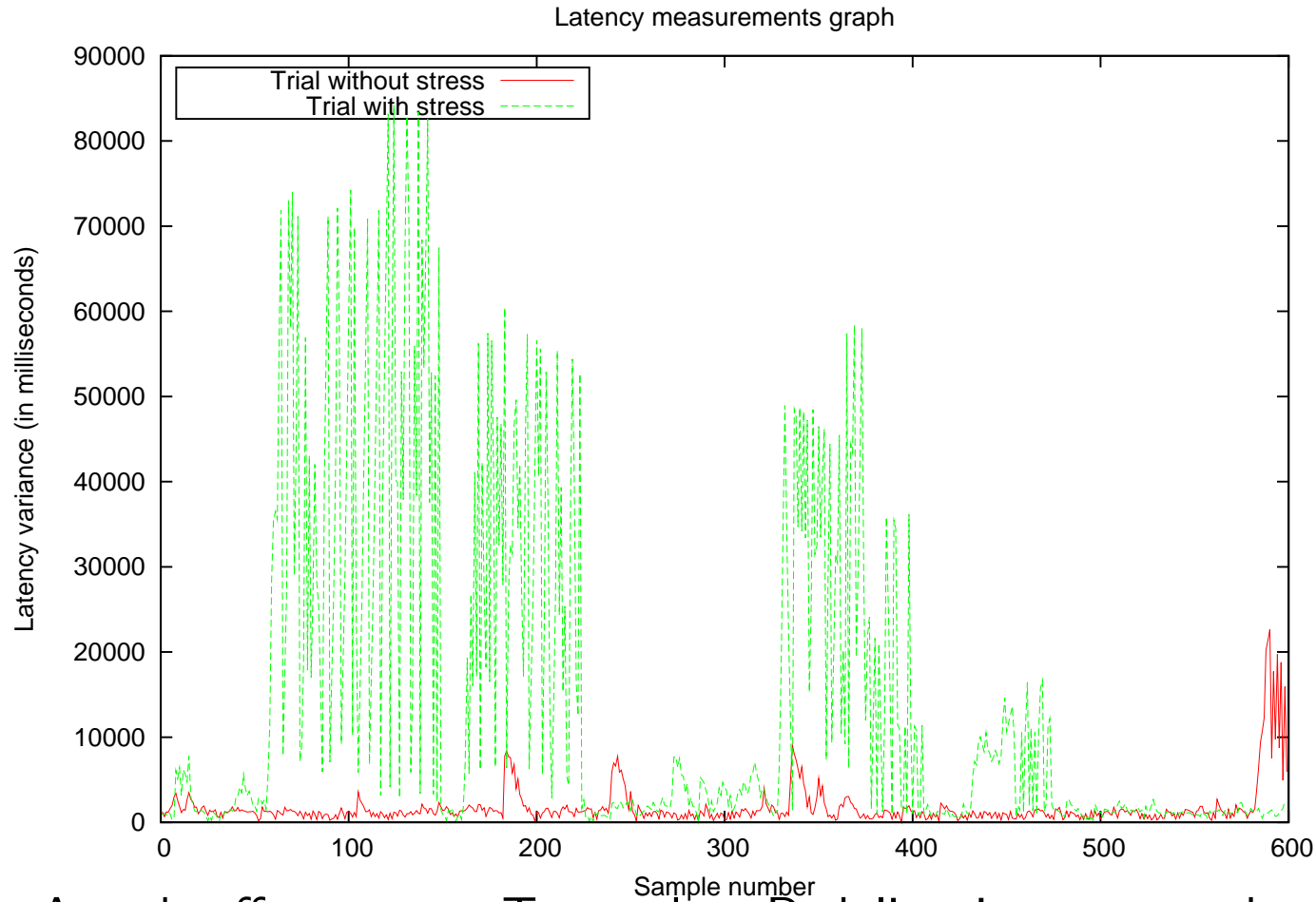


Figure 13: Attack effects on a Tor node. Red line is unstressed test, green is stressed test

Gathered Data Example (6/10)

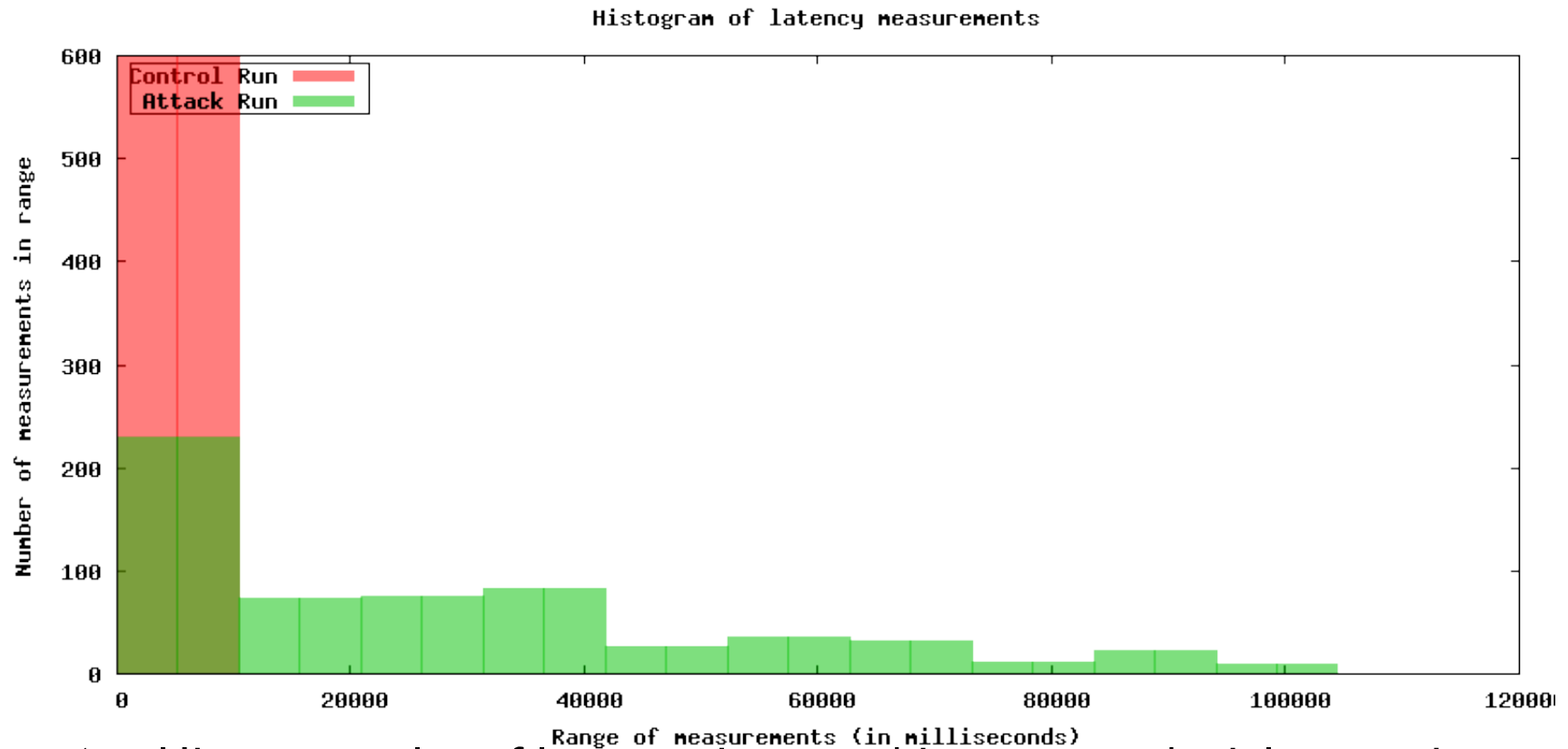


Figure 14: Histogram plot of latency times, red is unstressed trial, green is stressed. Each histogram is from the same trials as the previous set of graphs

Gathered Data Example (7/10)

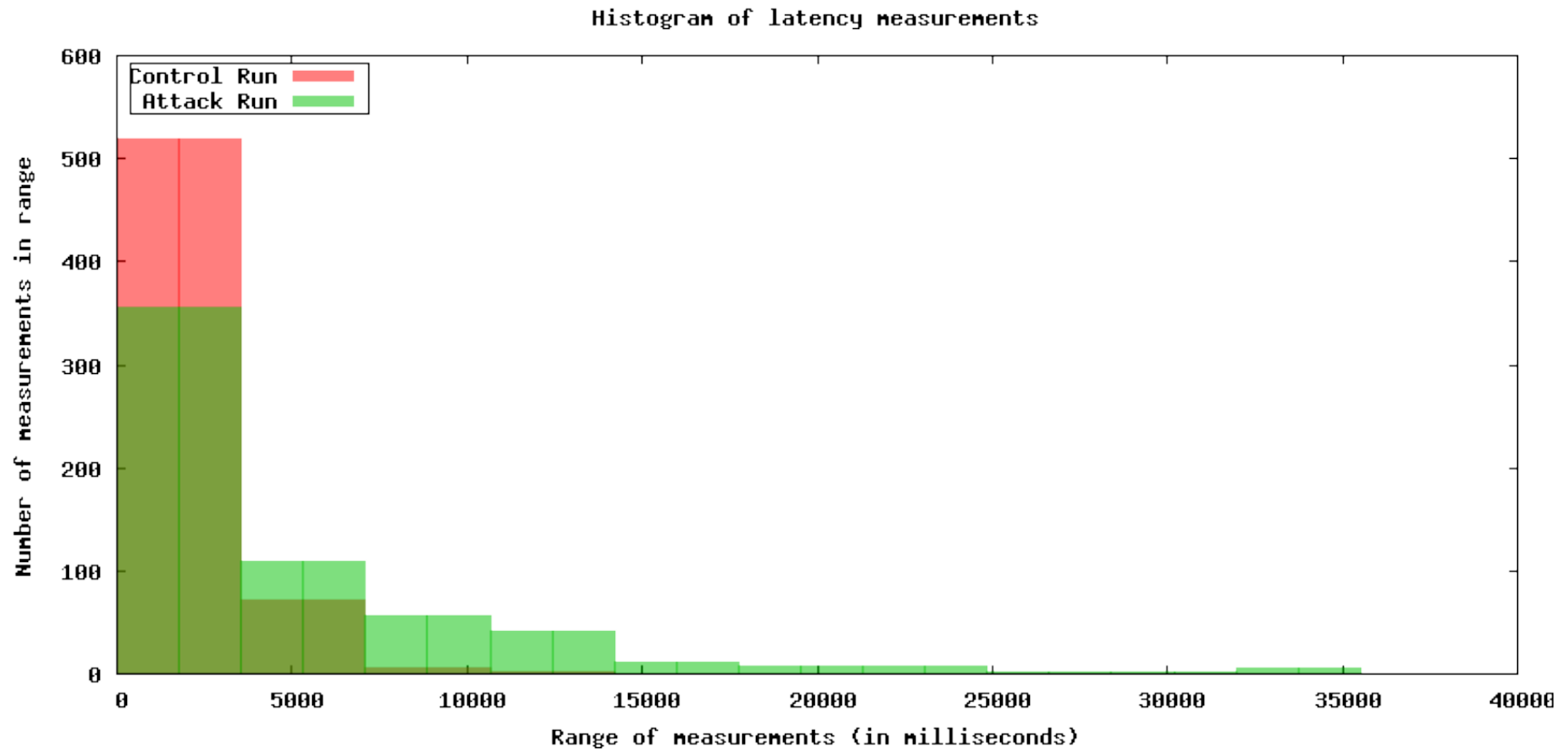


Figure 15: Histogram plot of latency times (same trials as previous graphs)

Gathered Data Example (8/10)

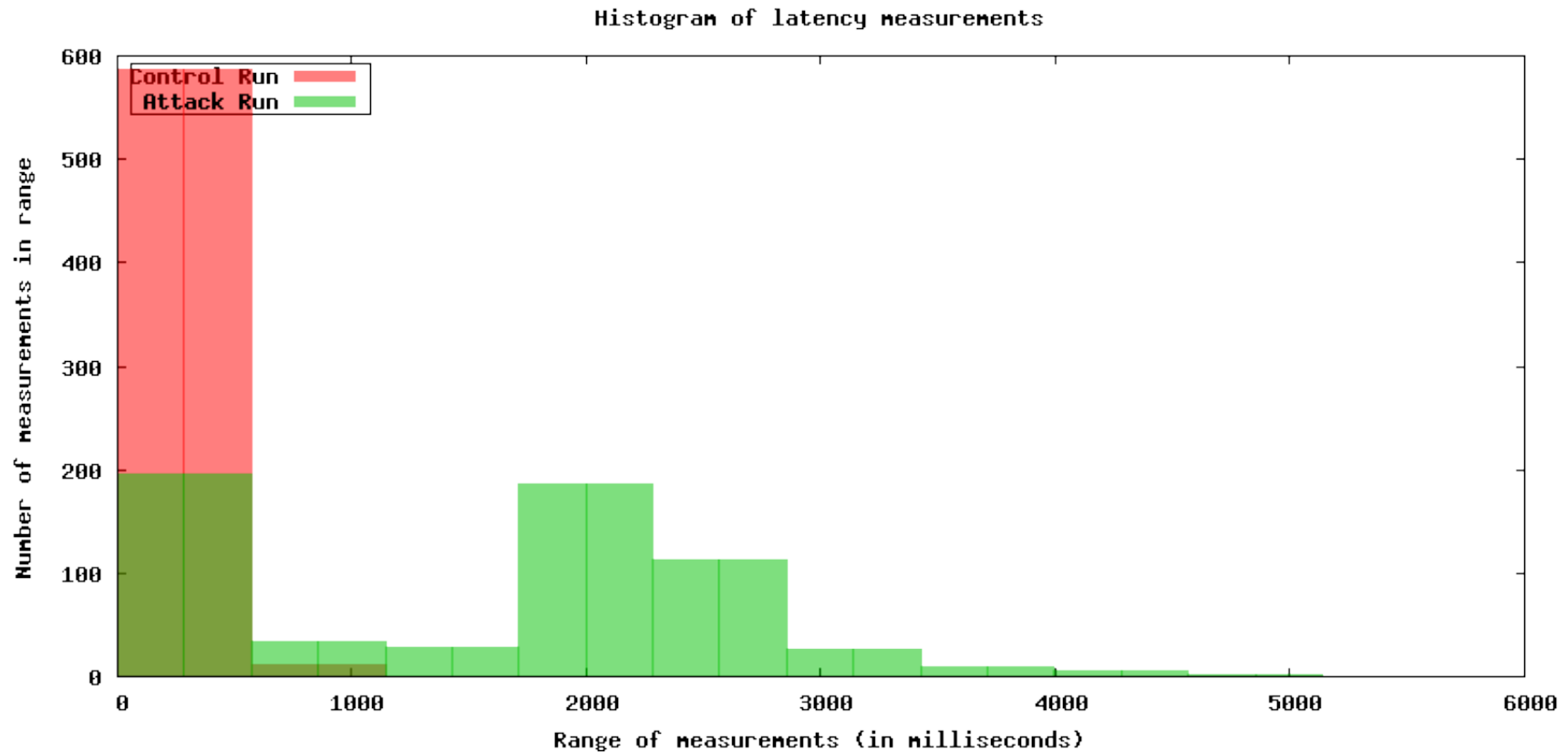


Figure 16: Histogram plot of latency times (same trials as previous graphs)

Gathered Data Example (9/10)

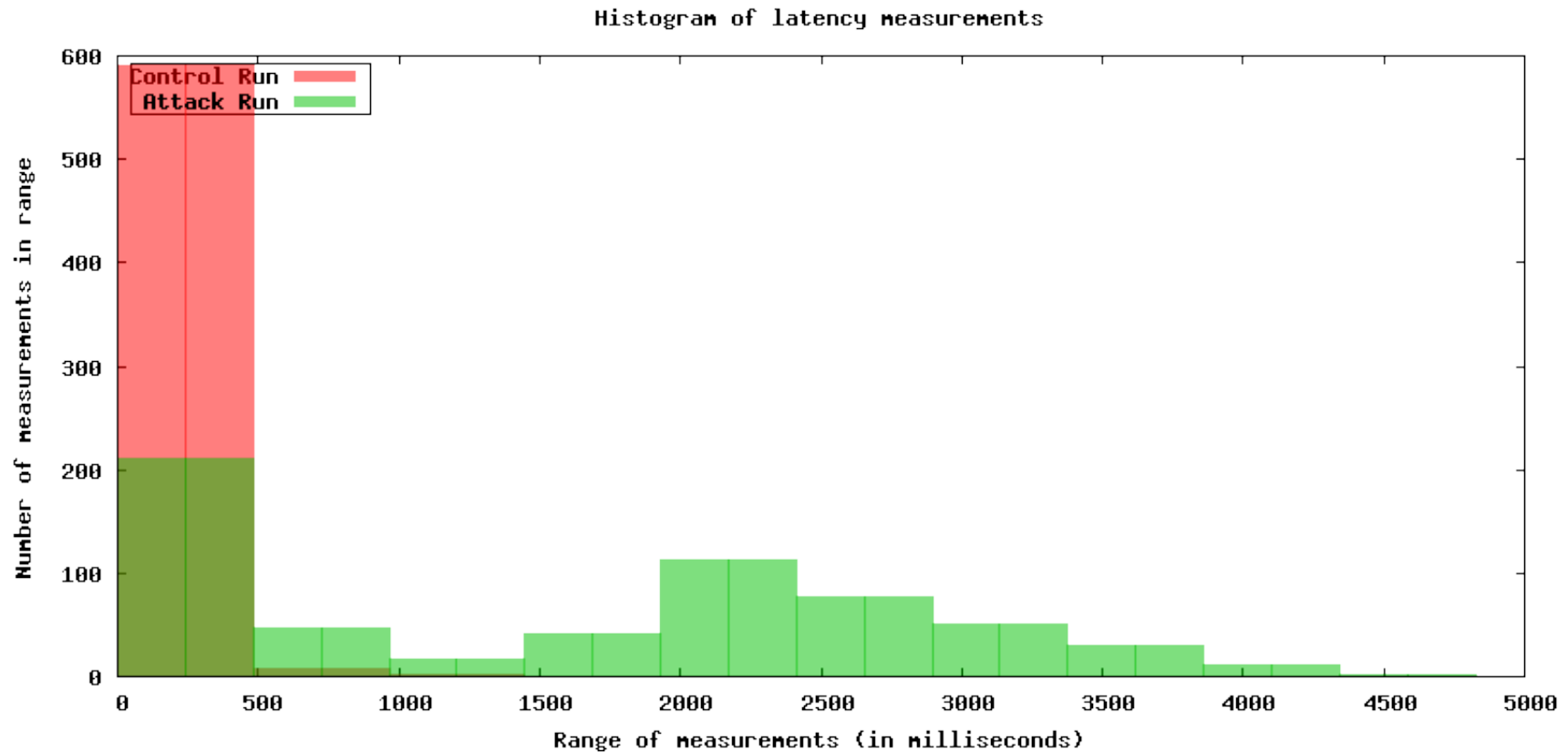


Figure 17: Histogram plot of latency times (same trials as previous graphs)

Gathered Data Example (10/10)

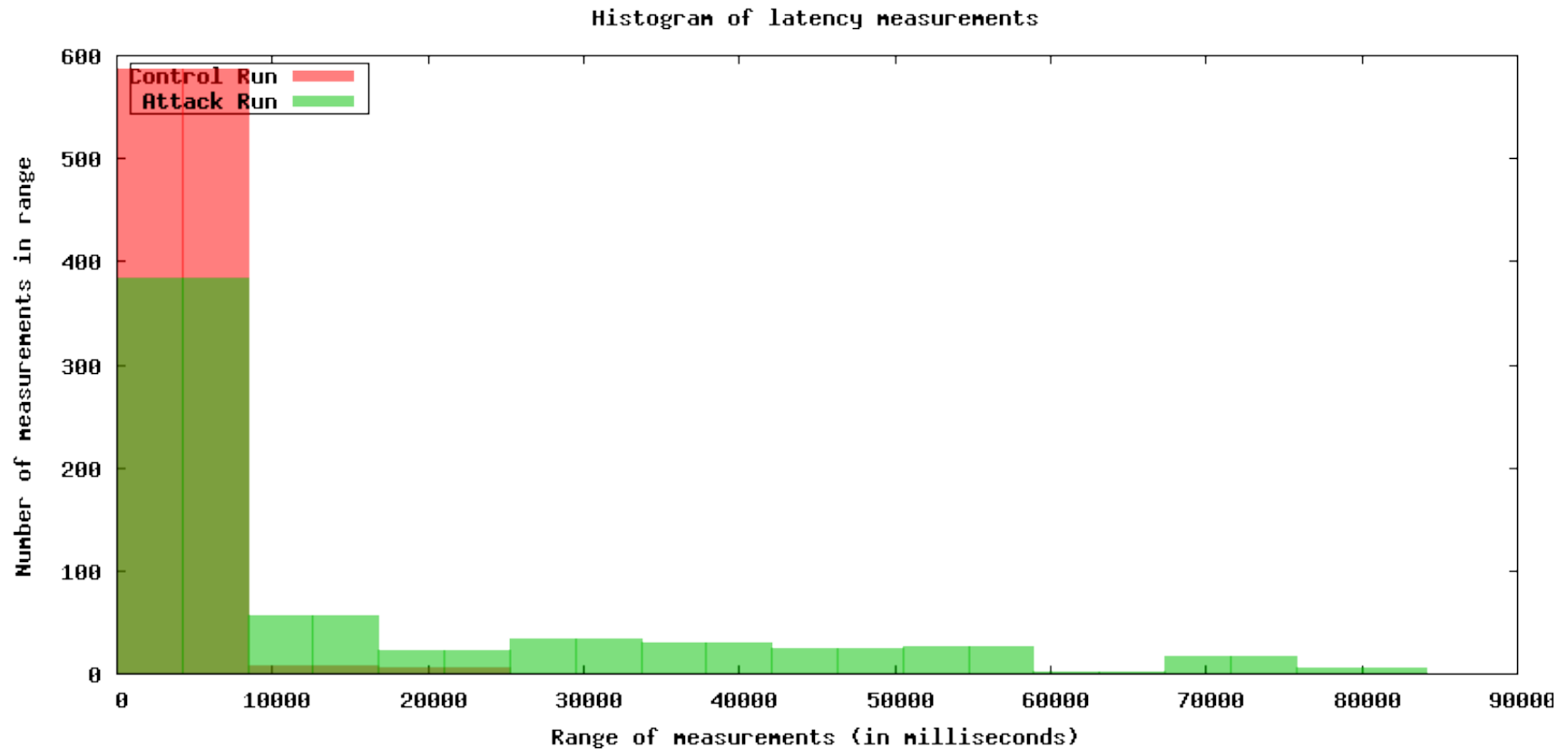


Figure 18: Histogram plot of latency times (same trials as previous graphs)

What We Actually Achieve

- We do identify the entire path through the Tor network (same result as Murdoch and Danezis)
- We do achieve this on the modern, current Tor network
- This means that if someone were performing this attack from an exit node, Tor becomes as effective as a one-hop proxy

Why Our Attack is Effective

- Since we run the exit router, only a single node needs to be found
- Our multiplication of bandwidth technique allows low bandwidth connections to DoS high bandwidth connections (solves common DoS limitation)

Fixes

- Don't use a fixed path length (or at least make it longer)
- Don't allow infinite path lengths
- Induce delays into connections (probably not going to happen)
- Monitor exit nodes for strange behavior (been done somewhat)
- Disable JavaScript in clients
- Use end-to-end encryption

Attack Improvements/Variants

- Use meta refresh tags for measurements instead of JavaScript
- Parallelize testing (rule out multiple possible first nodes at once)
- Improved latency measures for first hop to further narrow possible first hops

Conclusion

- Current Tor implementation allows arbitrary length paths
- Current Tor implementation uses minimally short paths
- Arbitrary path lengths allow DoS
- DoS allows detection of significant changes in latency
- Significant changes in latency reveal paths used

Questions?

