Anonymity With Tor

The Onion Router

Nathan S. Evans[1] Christian Grothoff[1]

[1]Technische Universität München

July, 8 2009

Overview

- What is Tor?
- Motivation
- Background Material
- How Tor Works
- Hidden Services
- Attacks
- Specific Attack
- Summary

What is Tor?

- Tor is a *P2P network* of Chaum inspired *low-latency mixes* which are used to provide *anonymous* communication between parties on the Internet.

What is Tor?

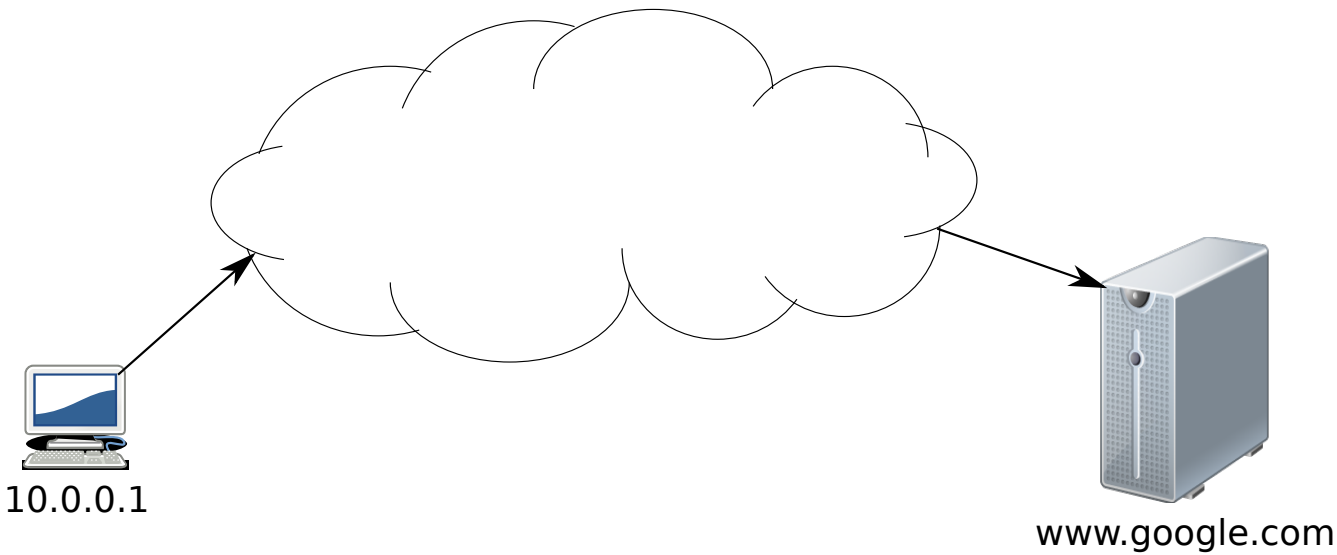
- Sender anonymity for low latency applications
- Common usage: Web browsing
  - Sender anonymity
  - Web server cannot identify client
- Advanced usage:
  - Hidden services (send/receive anonymity)
  - Filesharing
  - IRC
  - Any application that communicates using TCP

$\Rightarrow$ Tor provides users with a service that effectively hides their identity on the Internet.

Motivation

- Internet packets travel from A to B transparently
- A knows B, and B knows A (by IP address)
- Routers, etc. can determine that A and B are communicating
- This may reveal unintended information (e.g. person X's bank)
- Encryption
  - For example, TLS (HTTPS)
  - Provides *Data anonymity*
  - Does not hide routing information

Motivation - Routing Example

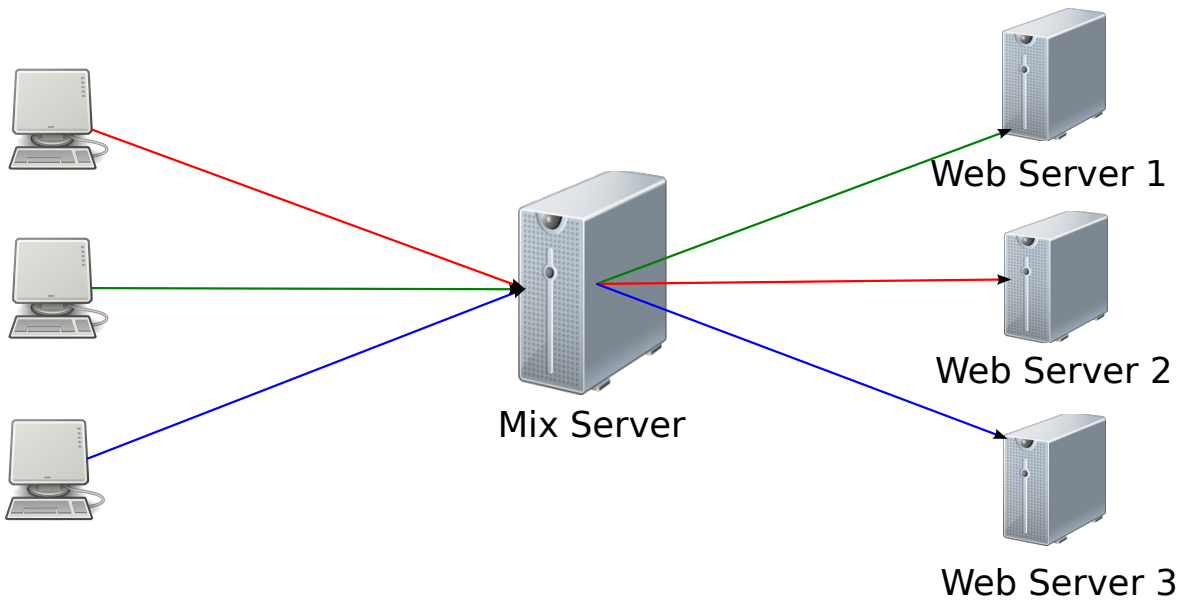


Motivation

- Who needs anonymous communication?
  - Political dissidents
  - Employees
  - Whistleblowers
  - Military, police, governments
  - Terrorists, criminals
  - You, me, everybody!
- There are times when people wish to communicate (or browse naughty web sites) without fear of reprisal, ridicule or punishment.

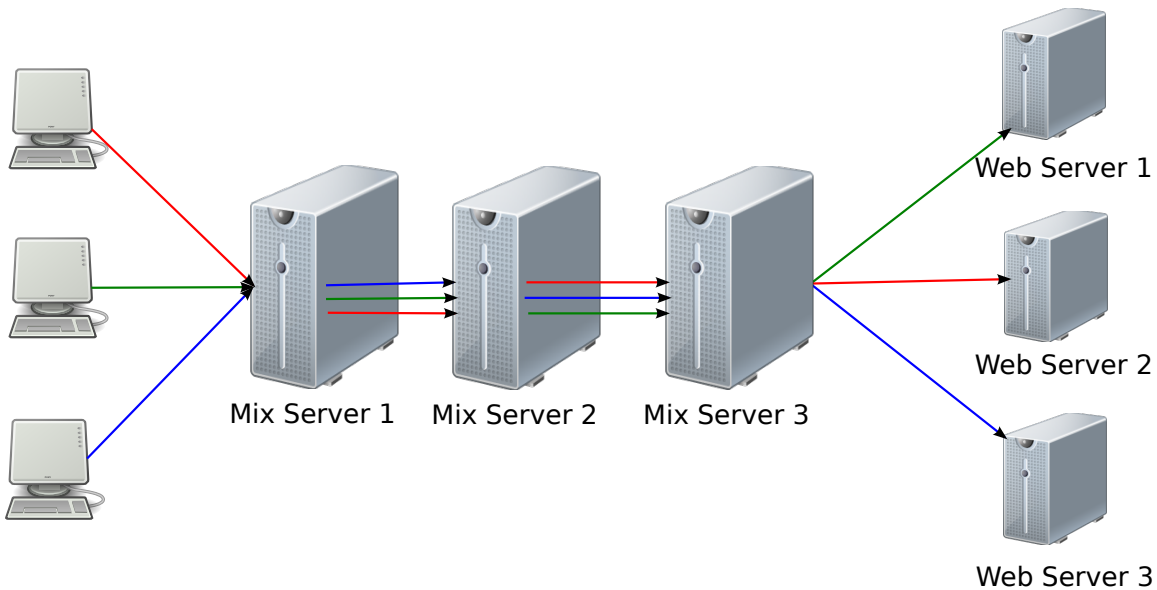
Background - Mixes

- A mix server takes requests from sources and sends them to destinations
- Hides the sender from the receiver
- Provides *Sender Anonymity*



Background - Mixes

- May delay or batch messages
- Usually used in series to provide better anonymity (why??)
- “Mix cascade”



Background - Proxies

- Similar to single mix server
  - Single server
  - Handles requests from clients
  - Returns results to clients
- Anonymizing proxy $\approx$ mix server
- Difference to mix server
  - Proxies generally used for different purposes
  - Caching responses for web pages
  - Corporate or educational blocking
  - Logging access

Background - Onion Routing

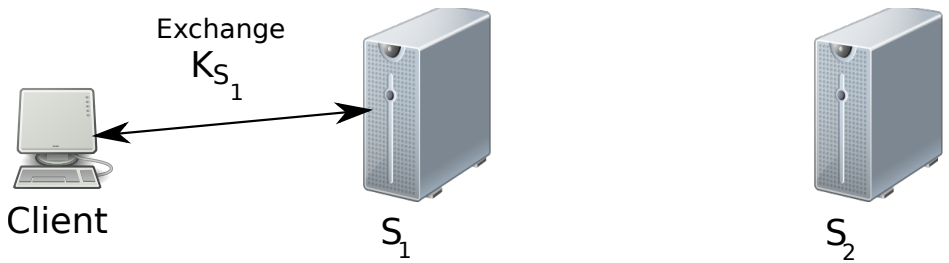
- Multiple mix servers
- Subset of mix servers chosen by initiator
- Chosen mix servers create "circuit"
  - Initiator contacts first server S_1, sets up symmetric key K_{S_1}
  - Then asks first server to connect to second server S_2; through this connection sets up symmetric key with second server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

Background - Onion Routing Example

- Assume $n = 2$, the initiator takes a message m to be sent along the circuit
- The initiator encrypts the message thusly, $K_{S_1}(K_{S_2}(m))$ and sends it to S_1
- S_1 decrypts the message with K_{S_1} and sends the result to S_2
- S_2 decrypts the message revealing m
- S_2 then performs whatever action is required on m

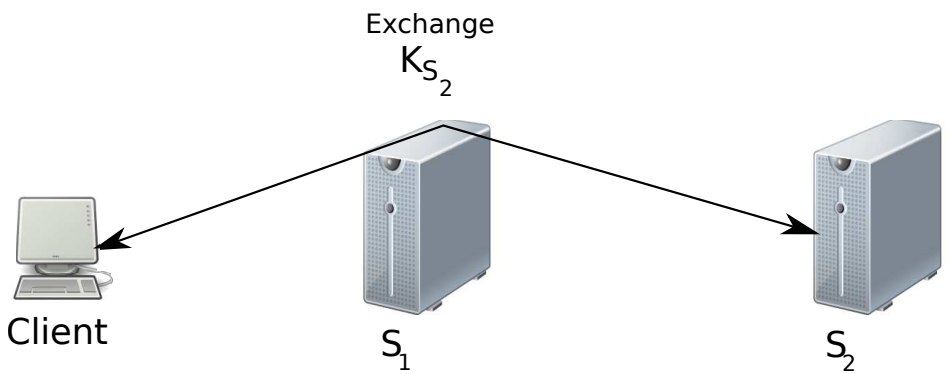
Background - Onion Routing Example

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



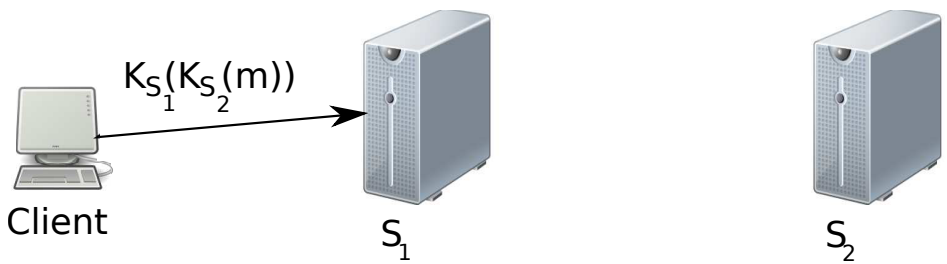
Background - Onion Routing Example

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



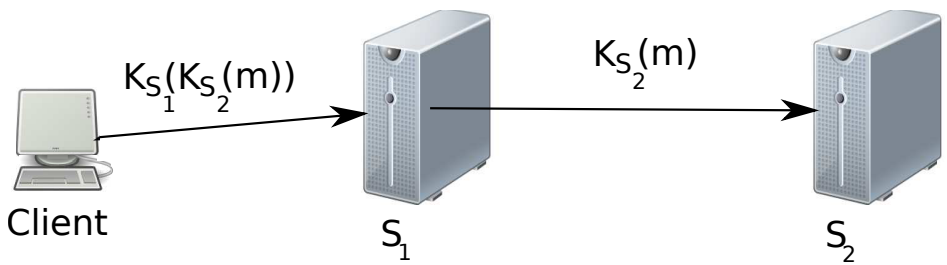
Background - Onion Routing Example

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



Background - Onion Routing Example

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

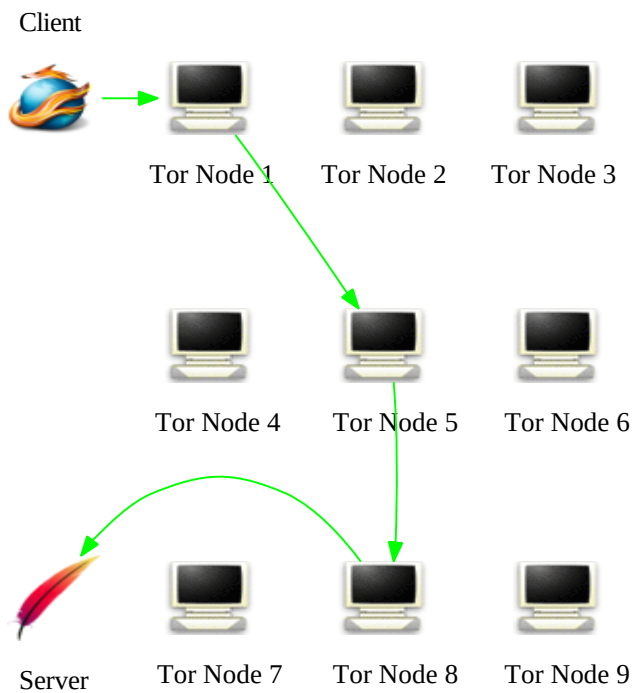


Tor - How it Works

- 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 hard coded at length three

Tor - How it Works - Example

- Example of Tor client circuit



Tor - How it Works - Servers

- Servers connected in "full mesh"
  - All servers exchange symmetric keys
  - Allows fast sending between servers, regardless of which circuits
  - Allows combining of multiple messages with same next-hop
- New servers publish information to directory servers
- Once online for a certain period, they are added to the "live" list
- They are then available for use by clients

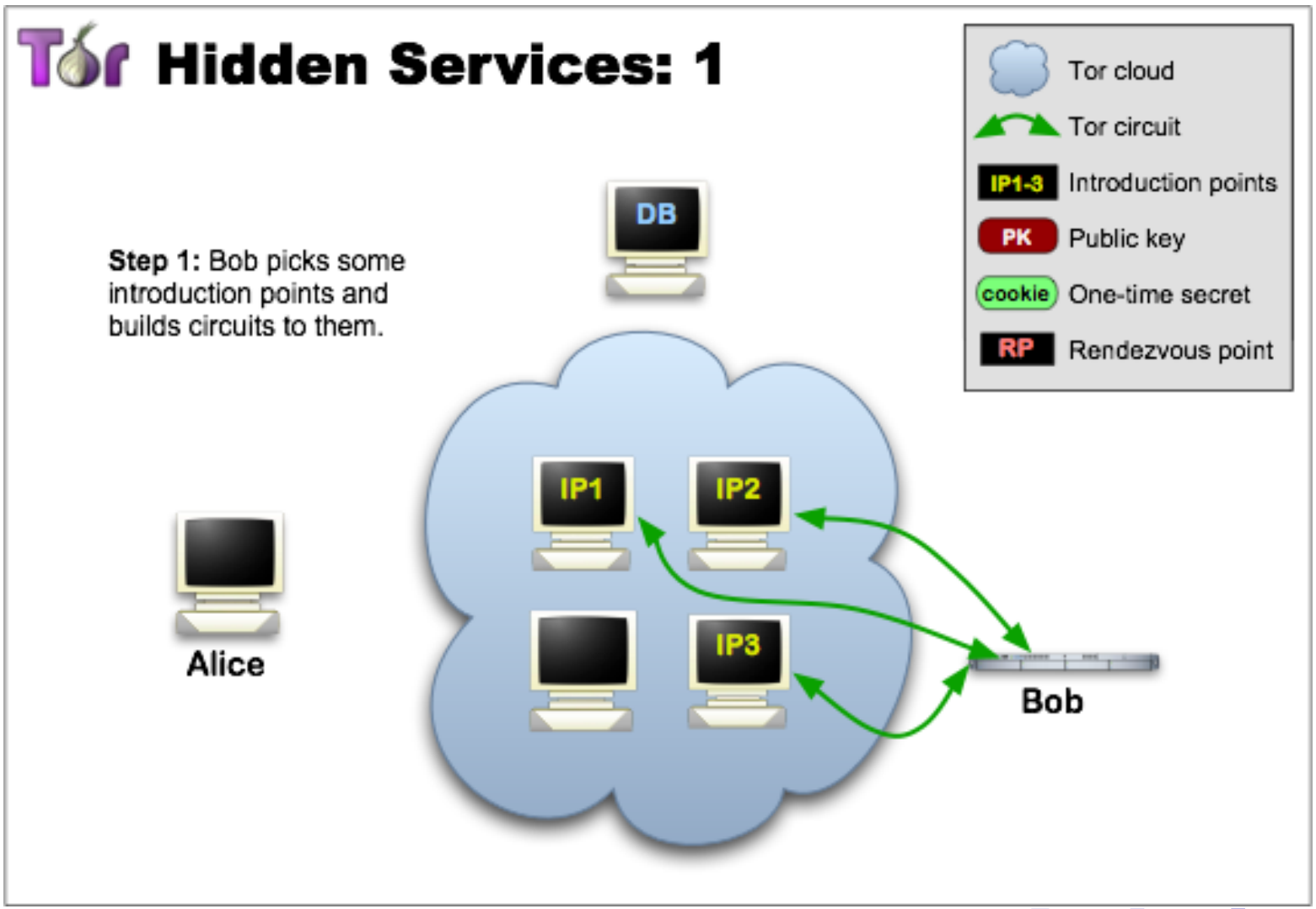
Tor - How it Works - Servers

- 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

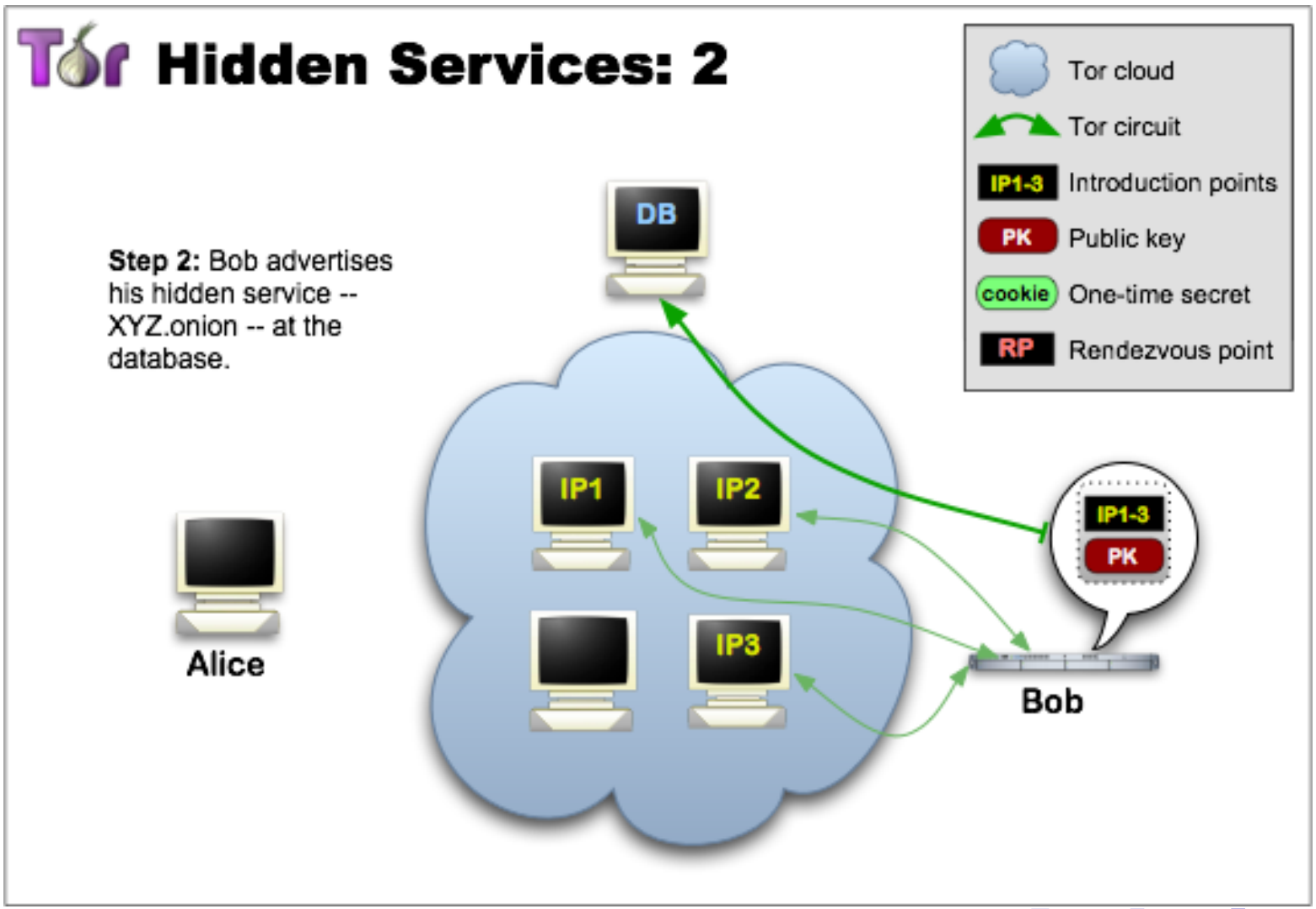
Hidden Services in Tor

- 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
- Uses a "Rendezvous point" to connect two Tor circuits
- Uses "Introduction points", which allow outside peers to contact hidden server (while keeping it hidden)
- Publishes Intro. point addresses to "Lookup server"
- Client gets Introduction point address from lookup server, sends random rendezvous point to hidden server
- Data travels a total of 7 hops (once established)

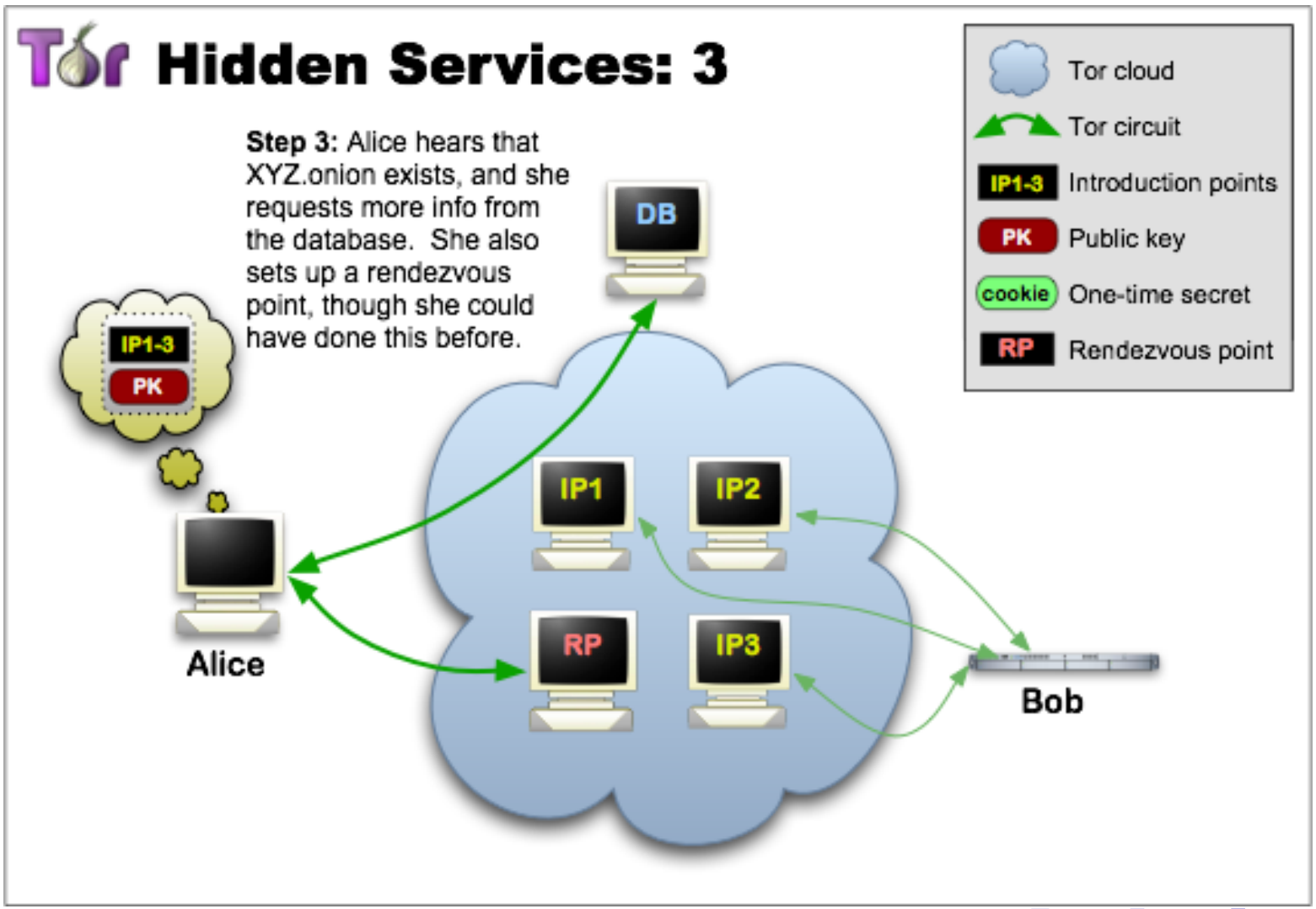
Hidden Services Example 1



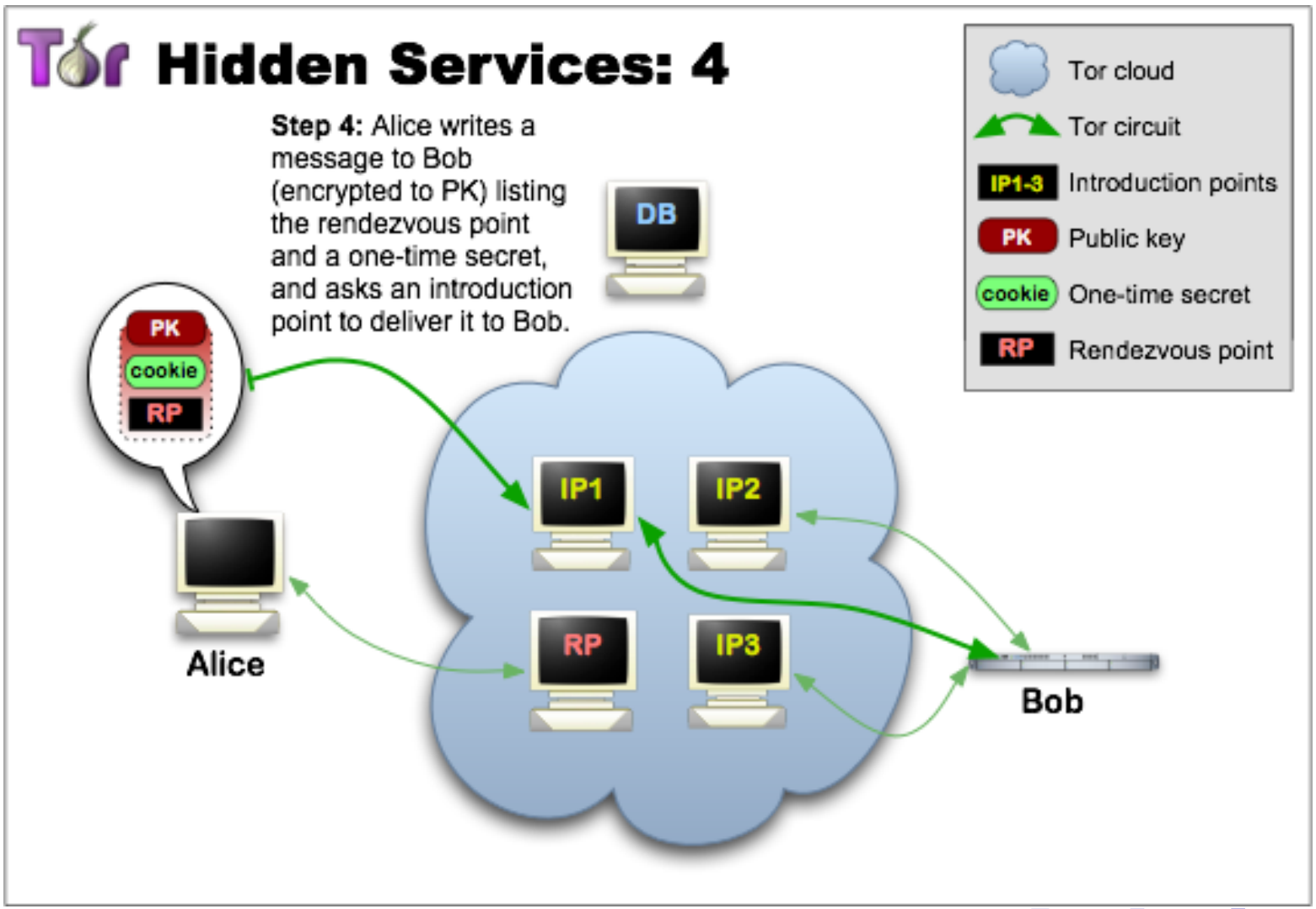
Hidden Services Example 2



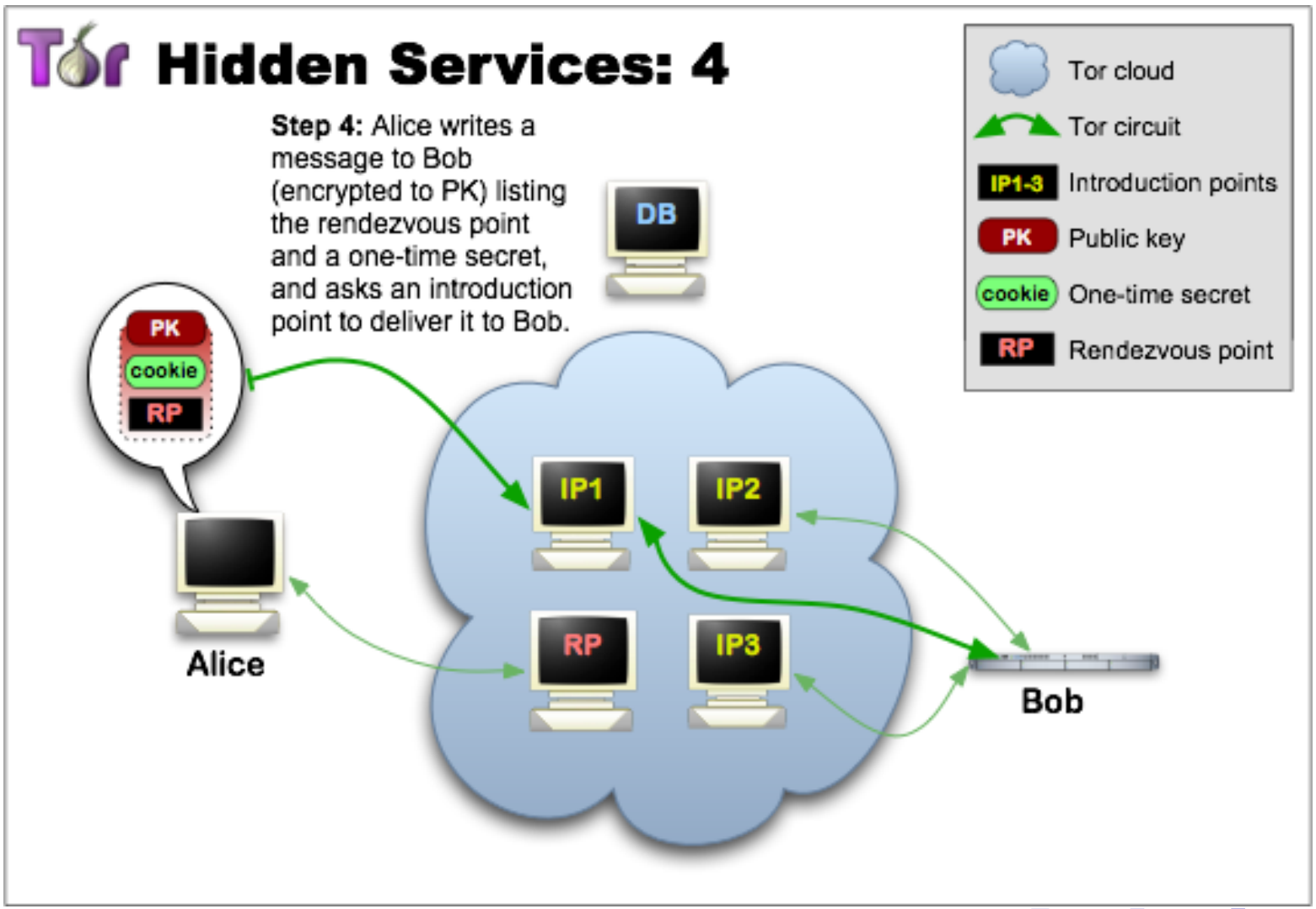
Hidden Services Example 3



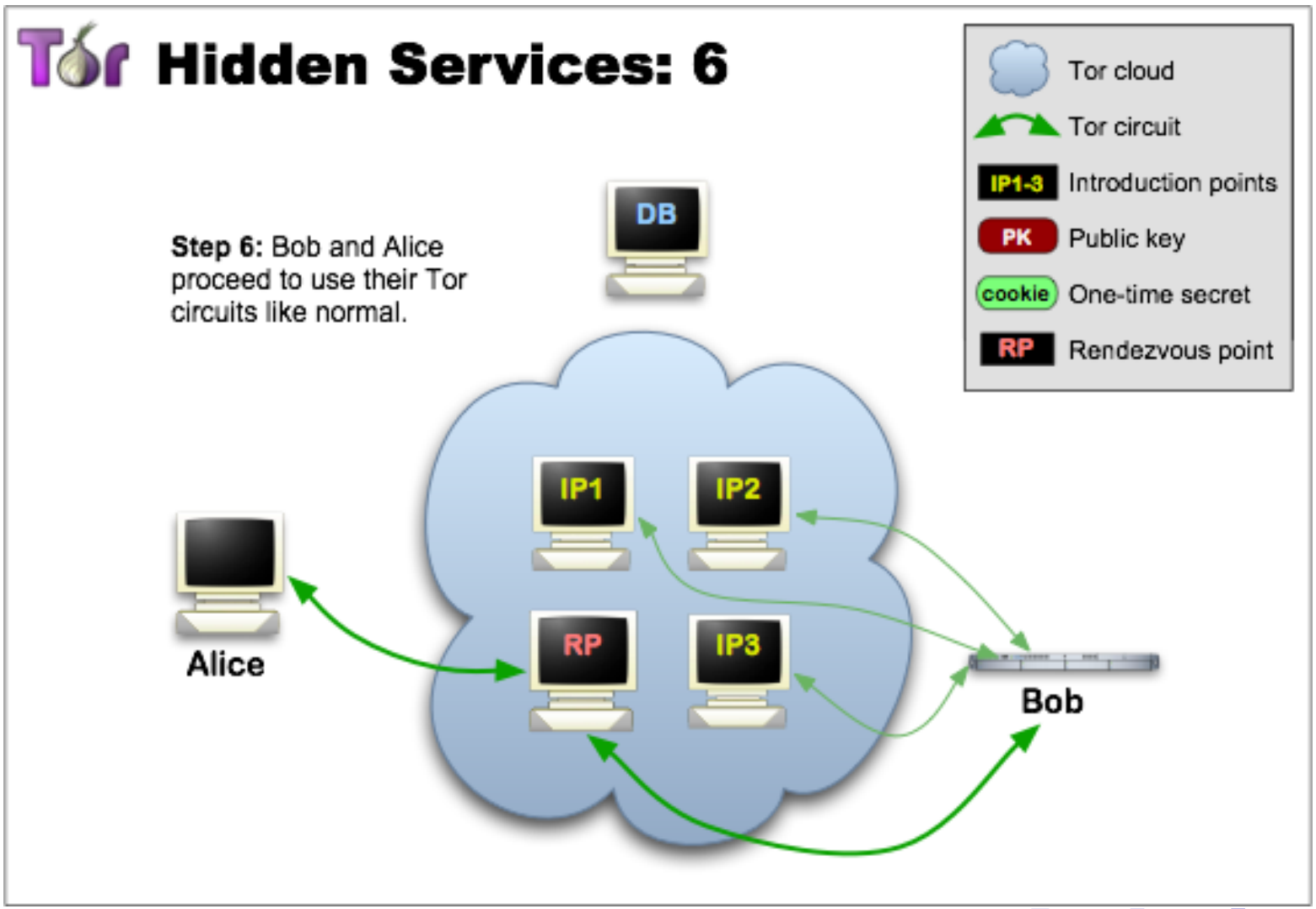
Hidden Services Example 4



Hidden Services Example 5



Hidden Services Example 6



Types of Attacks on Tor

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

Why attack Tor?

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

Our results:

All the Tor nodes involved in a circuit can be discovered, reducing Tor users level of anonymity and revealing a problem with Tor's protocol

Tor General Information

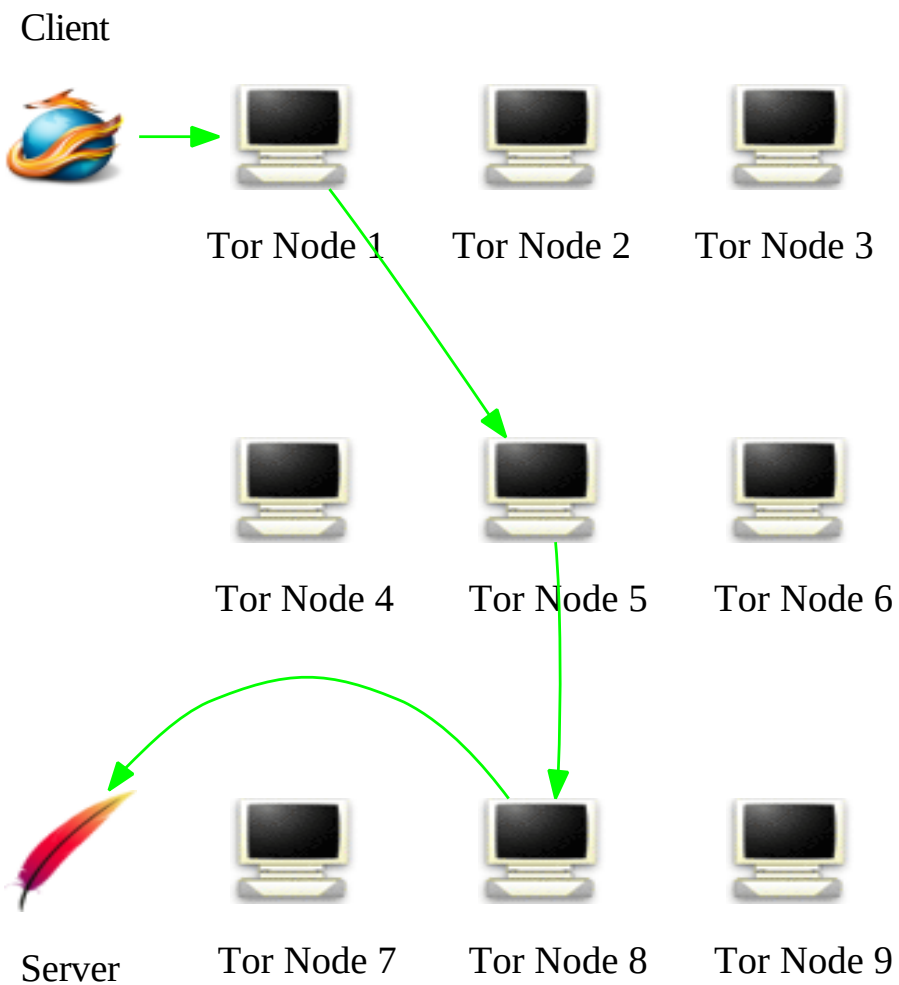
- Tor stands for "The onion router"
  - Encrypts data multiple times and is decrypted as it travels through the network a layer at a time: like peeling an onion
- Tor is a P2P network of mixes
- Routes data through network along a "circuit"
- Data is encrypted as it passes through nodes (until the last 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



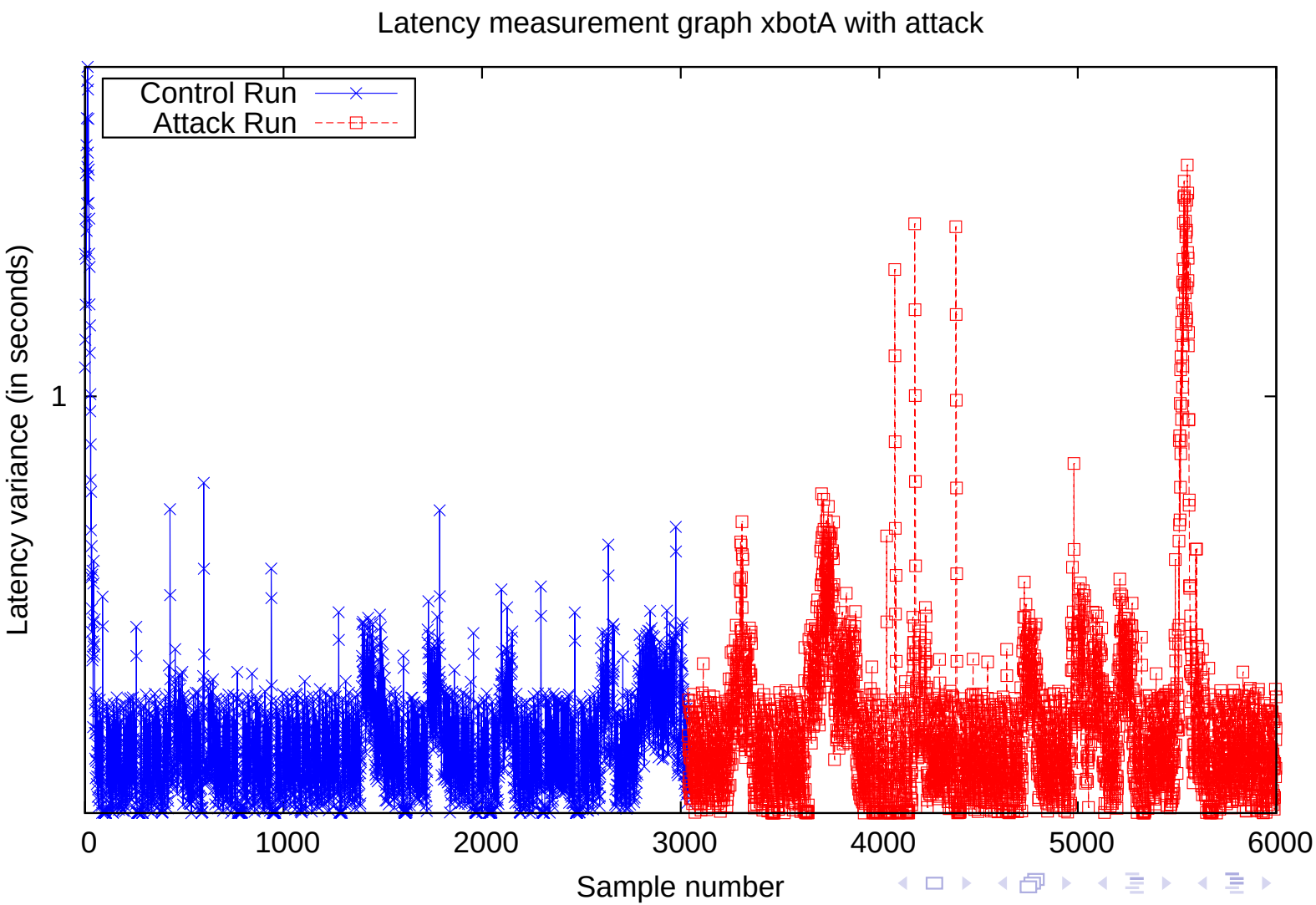
Previous work

- 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 reasonably well with the 13 Tor routers they used in 2005 (with 15% false negative rate)

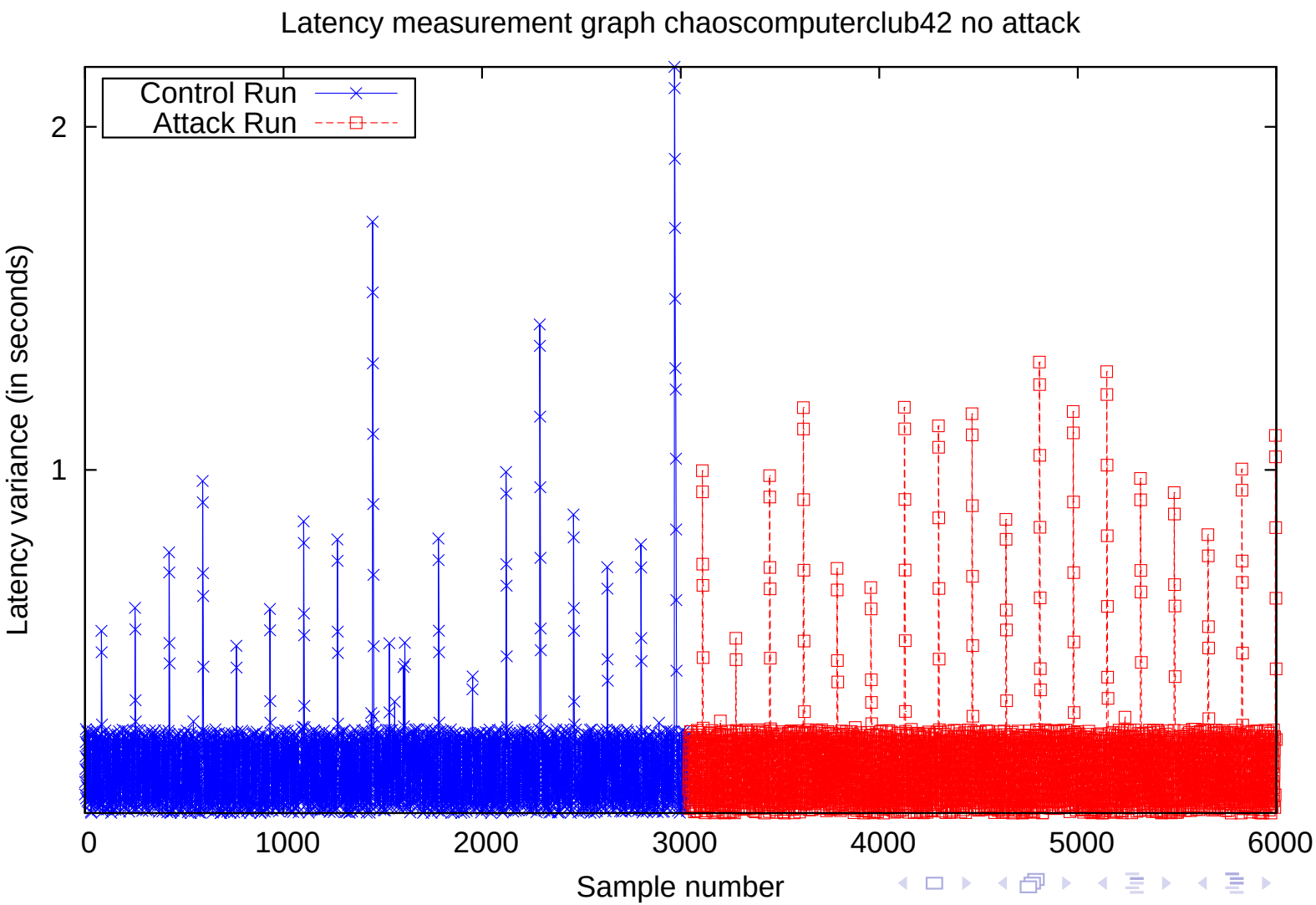
Problems With Previous Work

- Too inaccurate with today's 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

M and D Results - With Attack



M and D Results - Without Attack



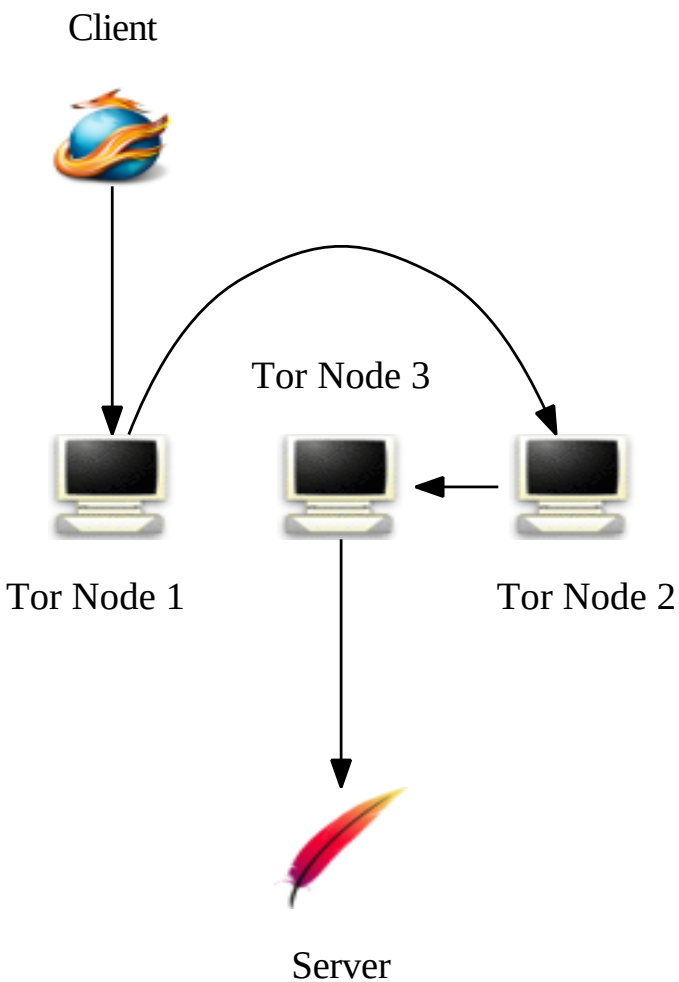
M and D Testing

- Used same statistical methods for correlation
- Used same source code for attacks
- In our tests, highest correlations seen with false positives
- Attack may be viable for some Tor nodes
- Improved statistical methods may improve false positives

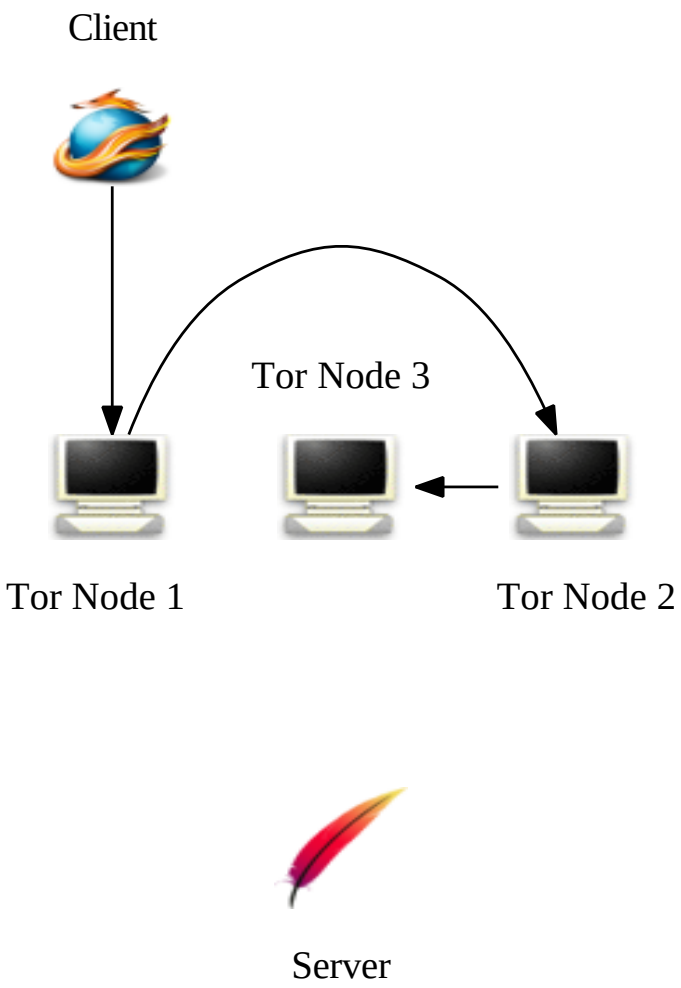
Our Basis for Deanonymization

- Target user is running Tor with privoxy with all the default settings
- 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

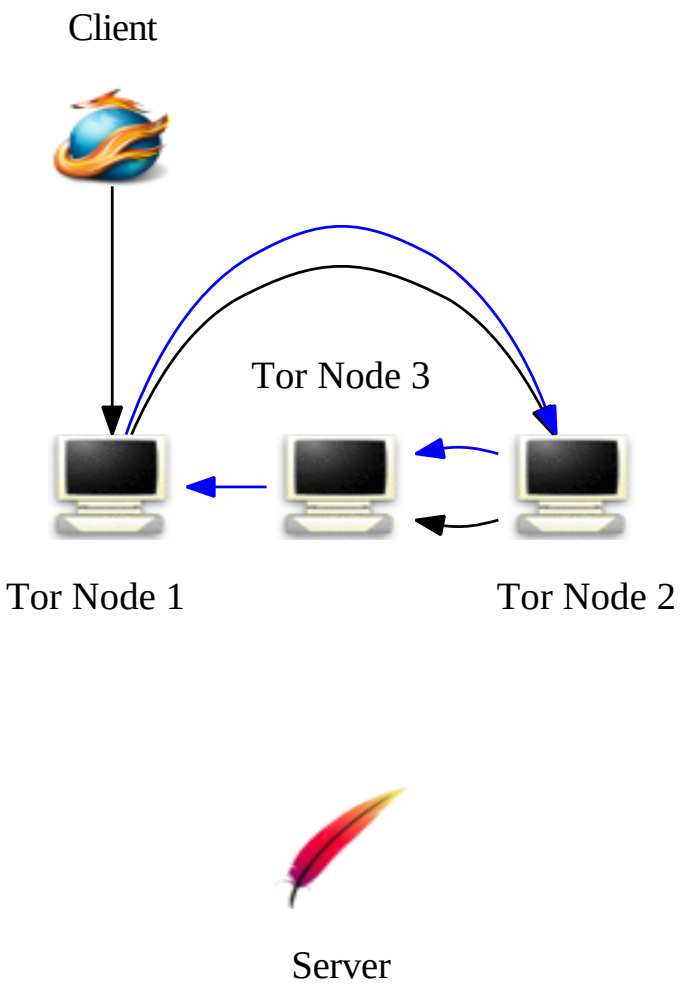
Regular Path Example



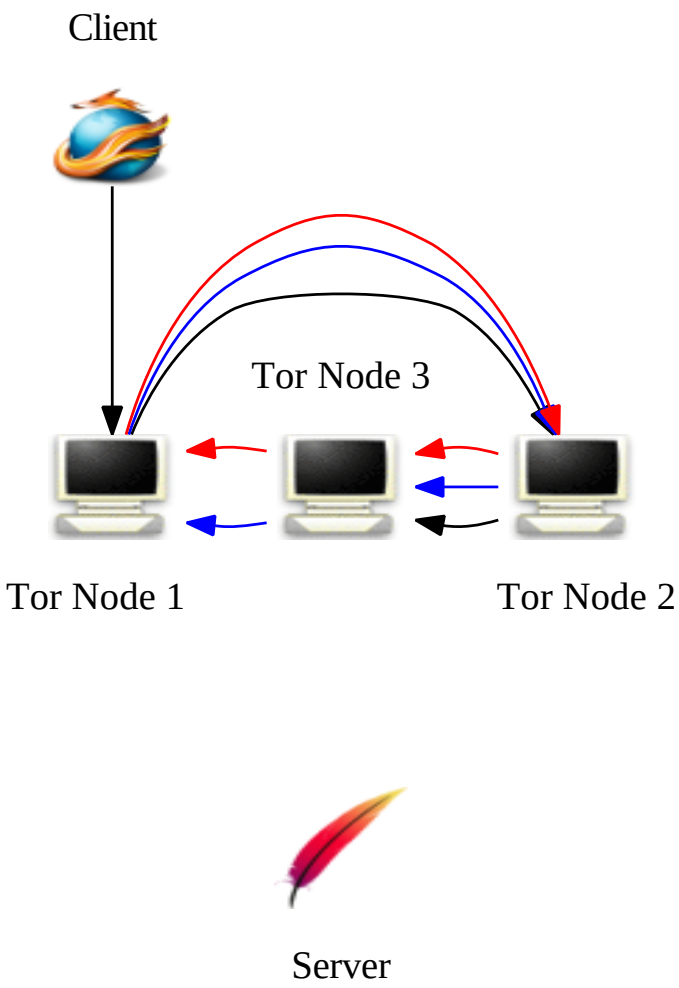
Circular Path Example 1/5



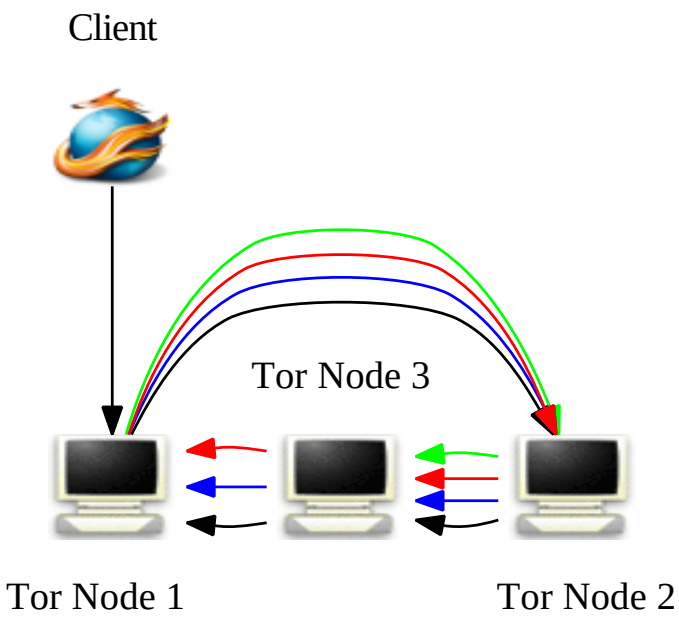
Circular Path Example 2/5



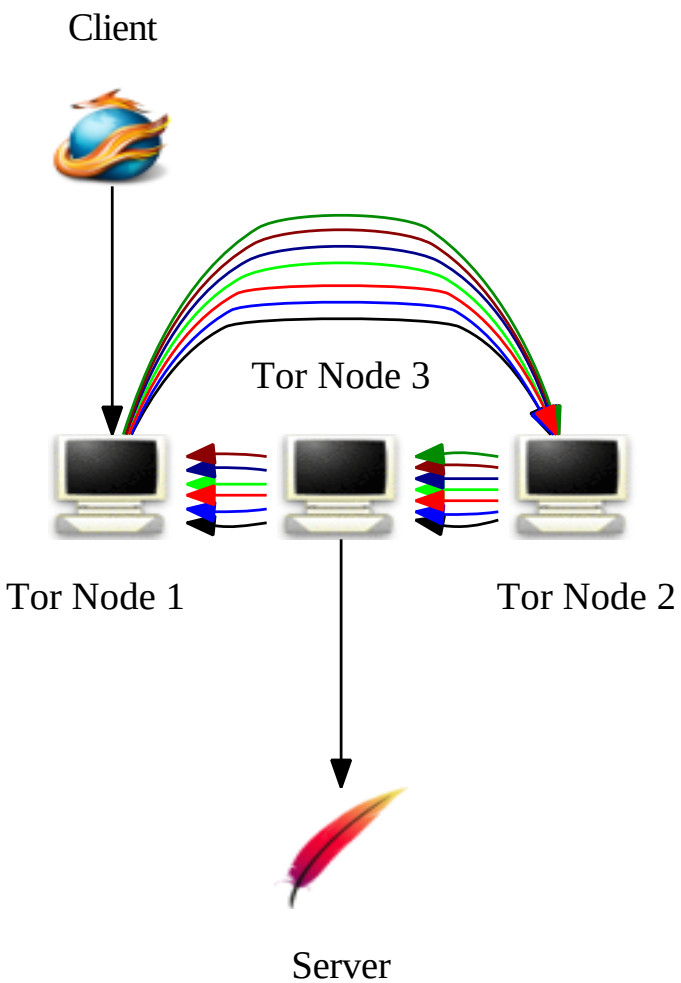
Circular Path Example 3/5



Circular Path Example 4/5



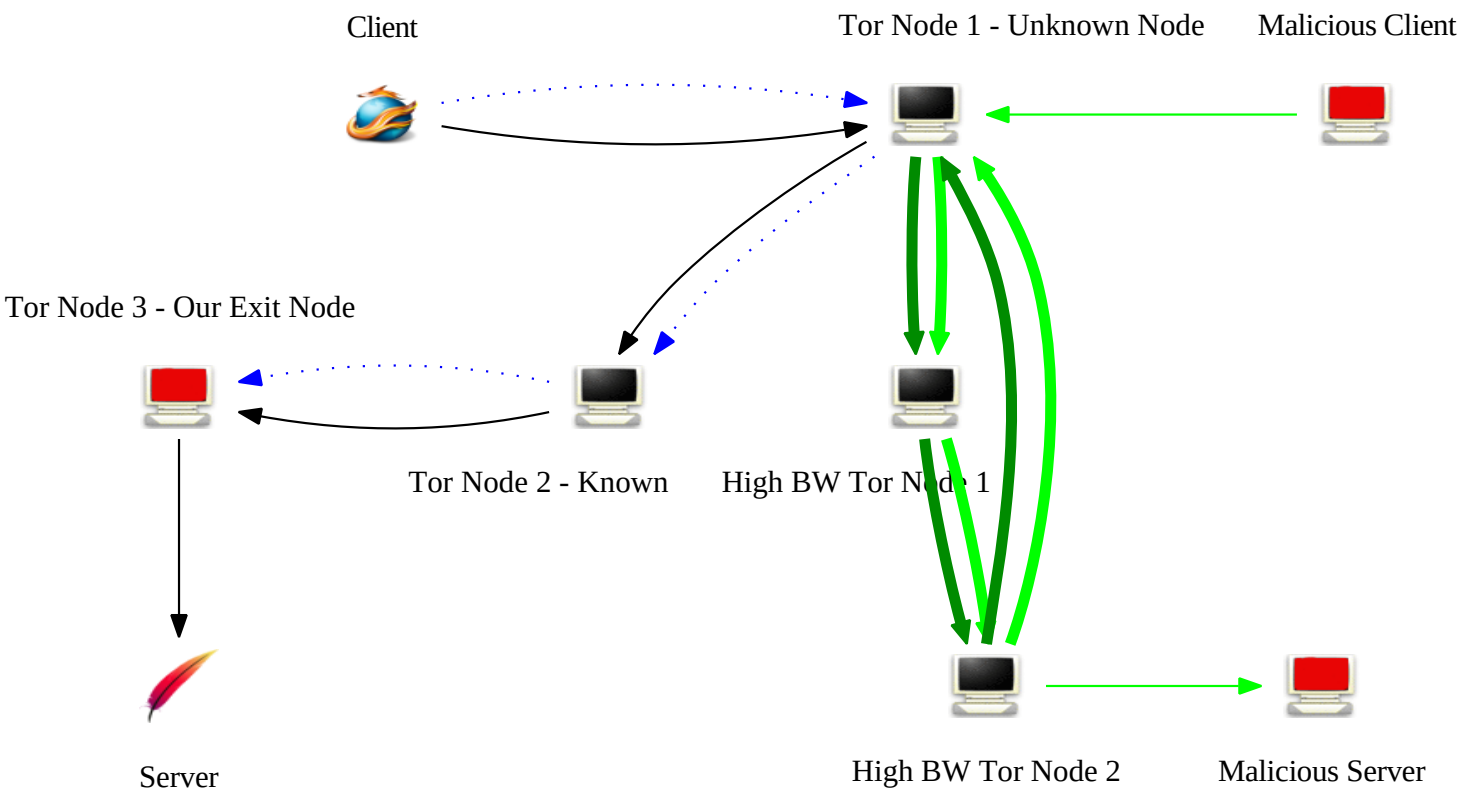
Circular Path Example 5/5



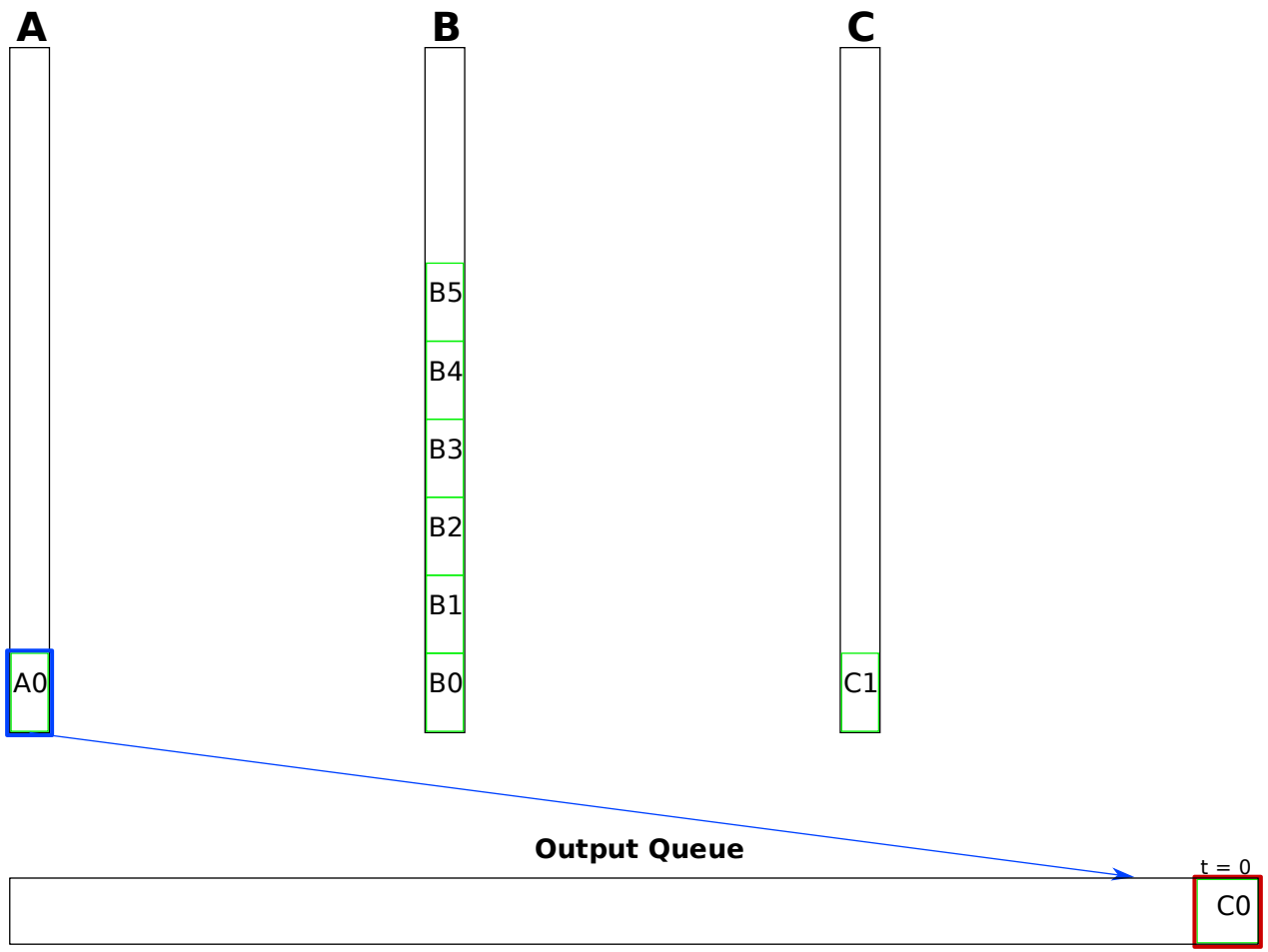
Attack Implementation

- 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, 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 *repeated* measurements, initial node has been found

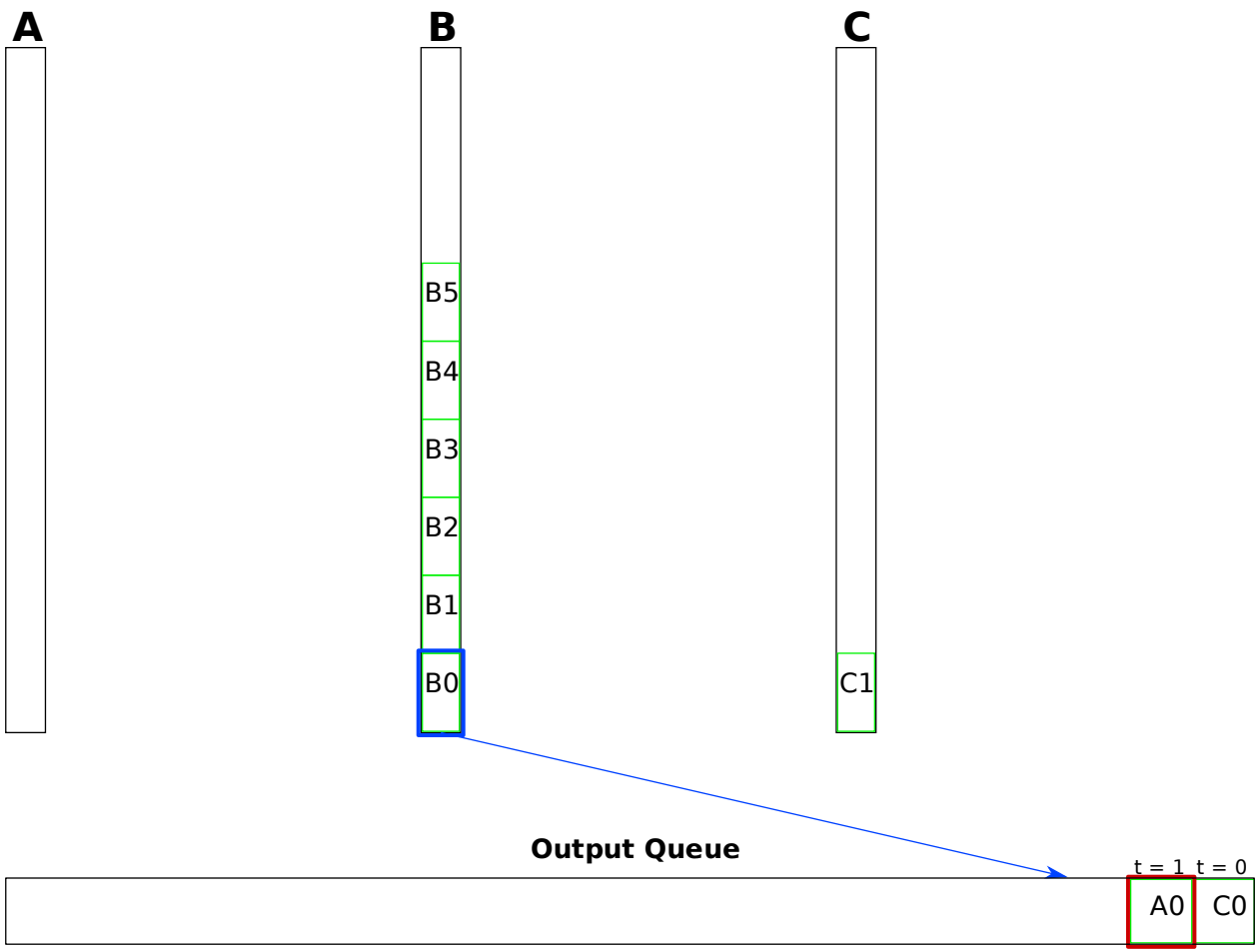
Attack Example



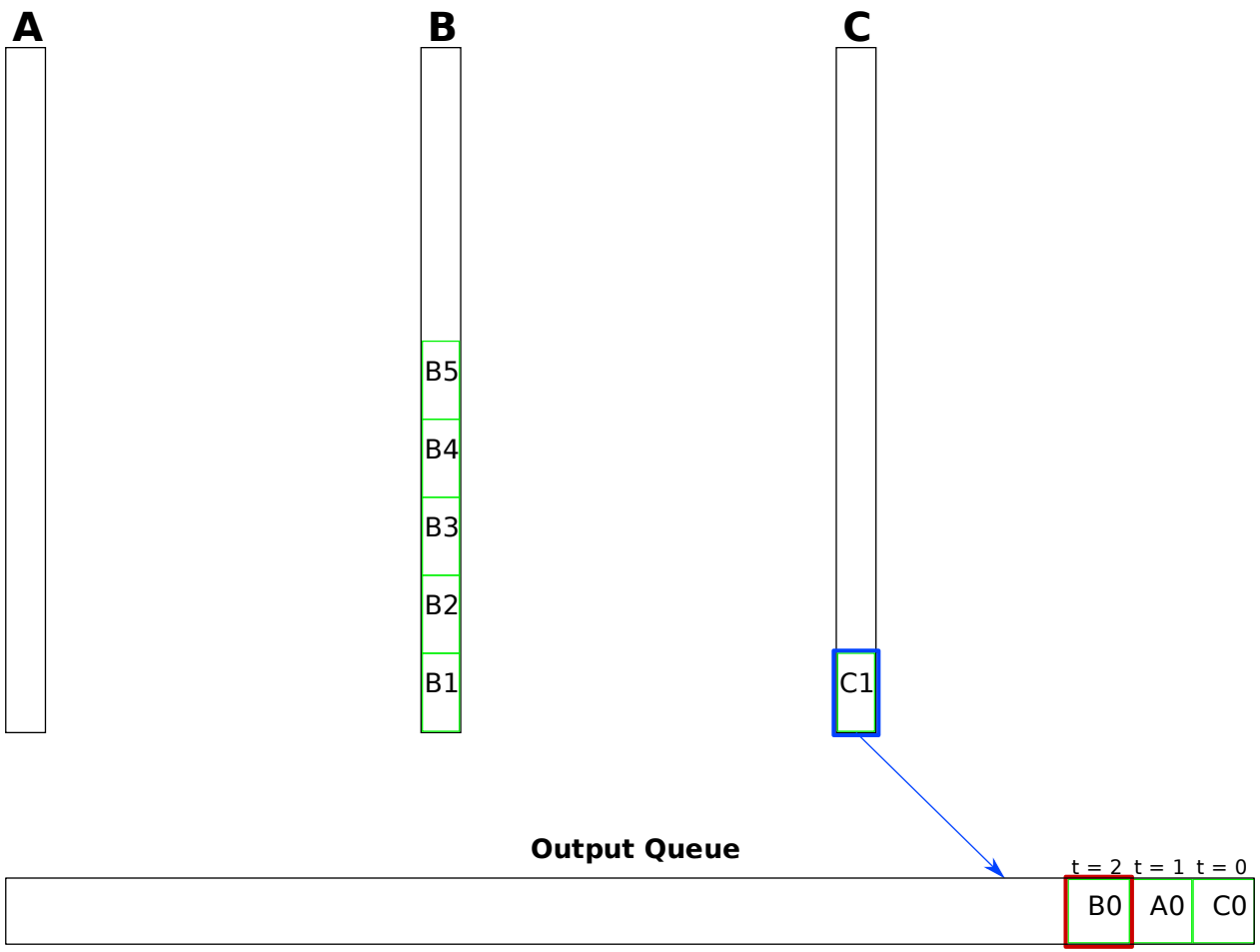
Queue example 1 (3 circuits)



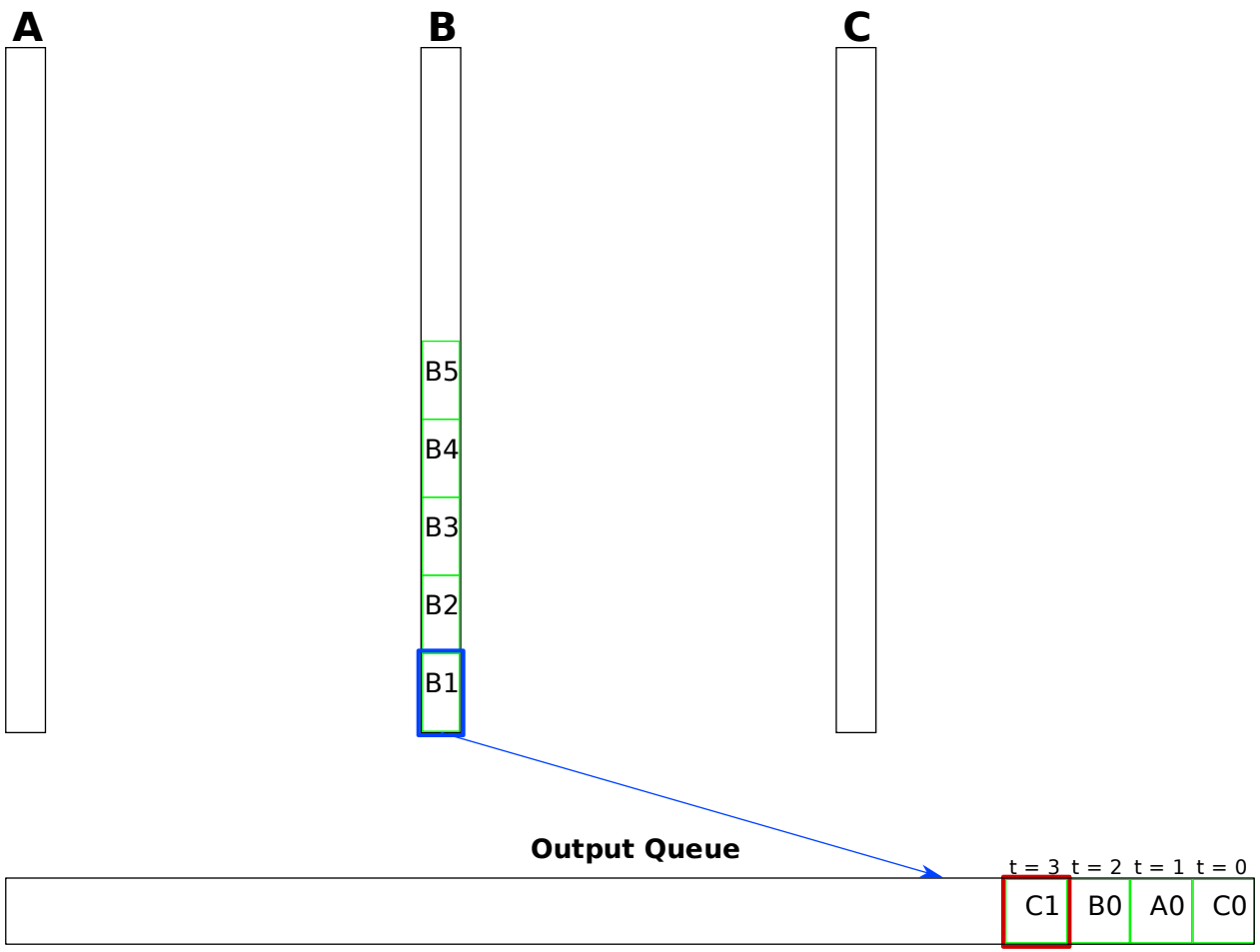
Queue example 2 (3 circuits)



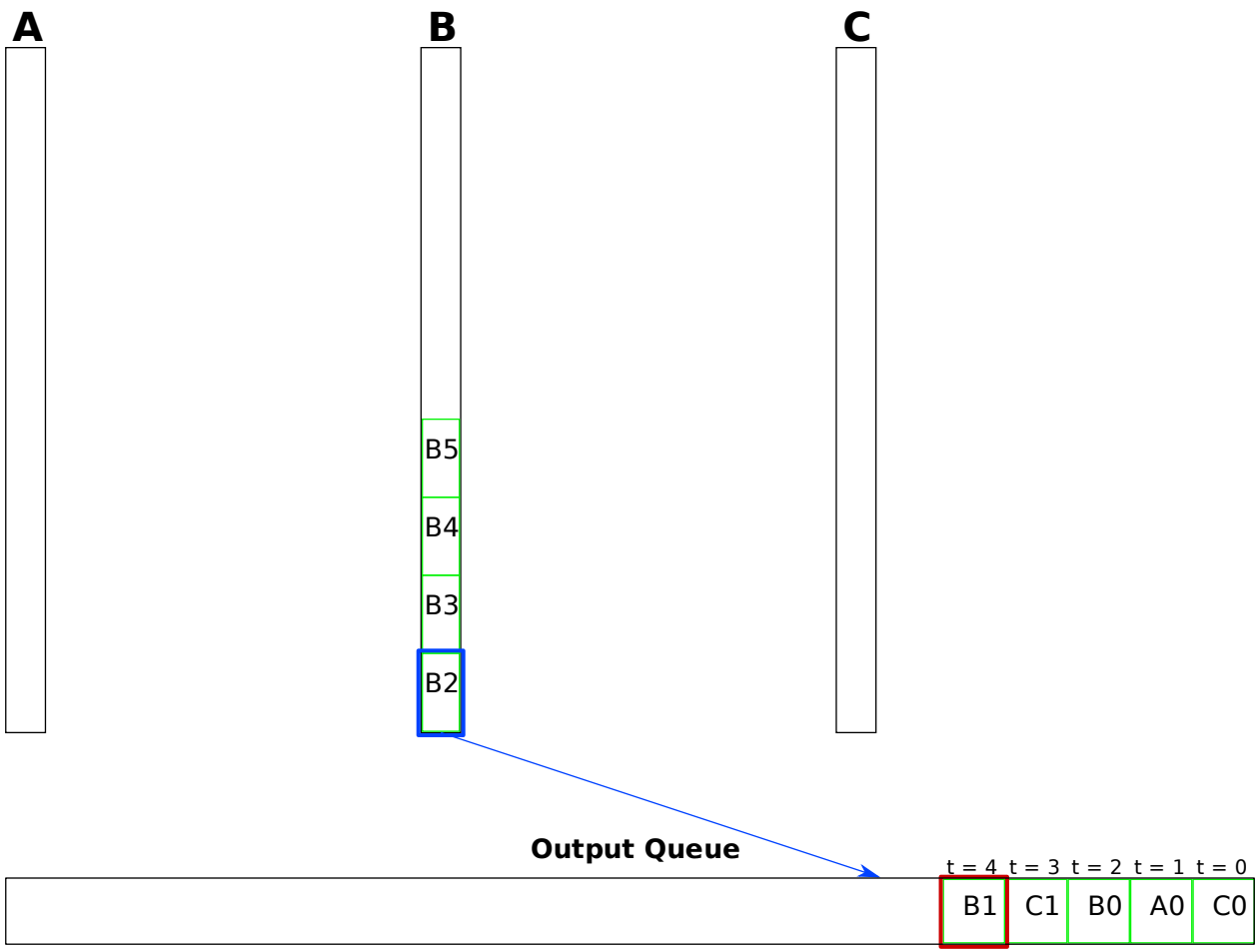
Queue example 3 (3 circuits)



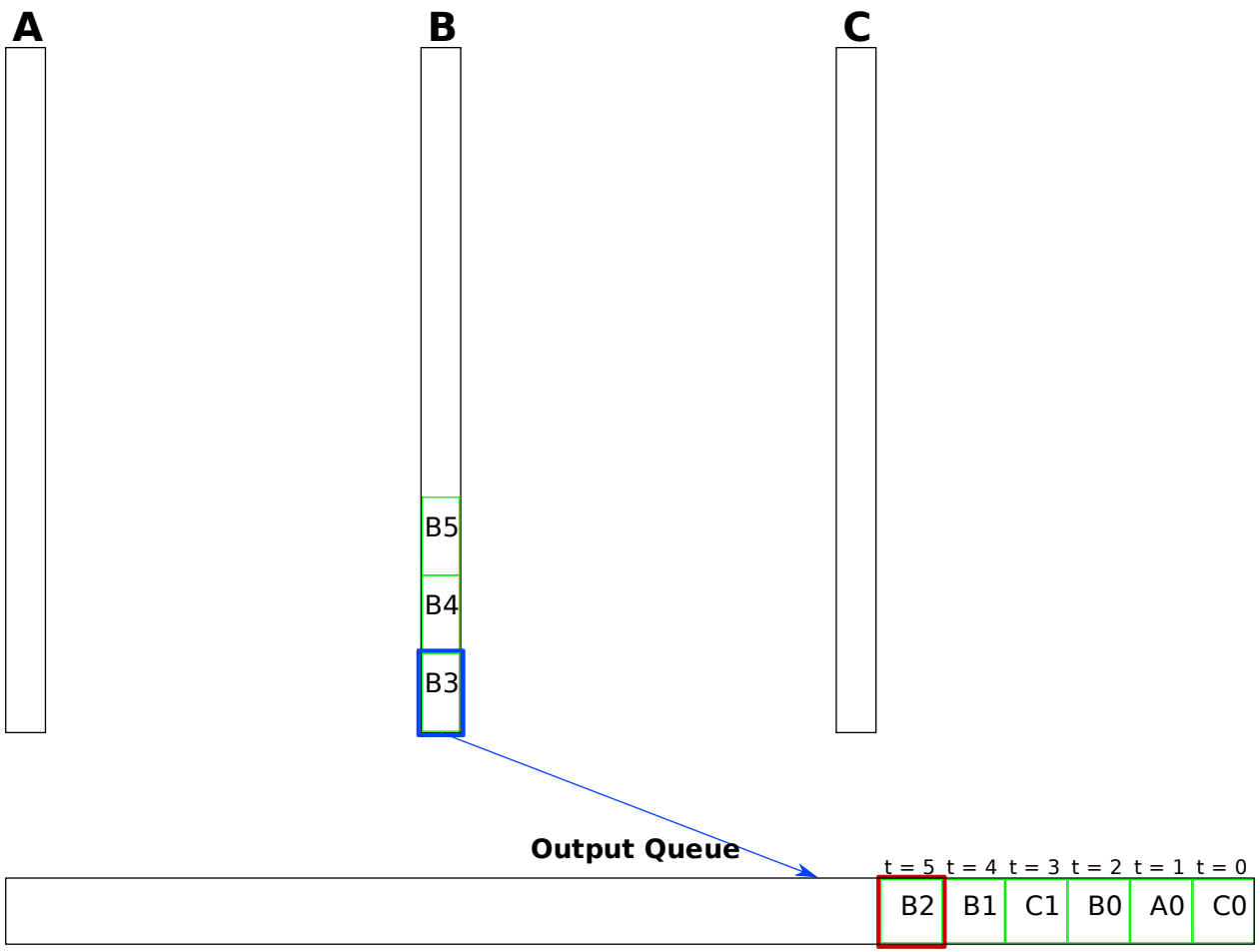
Queue example 4 (3 circuits)



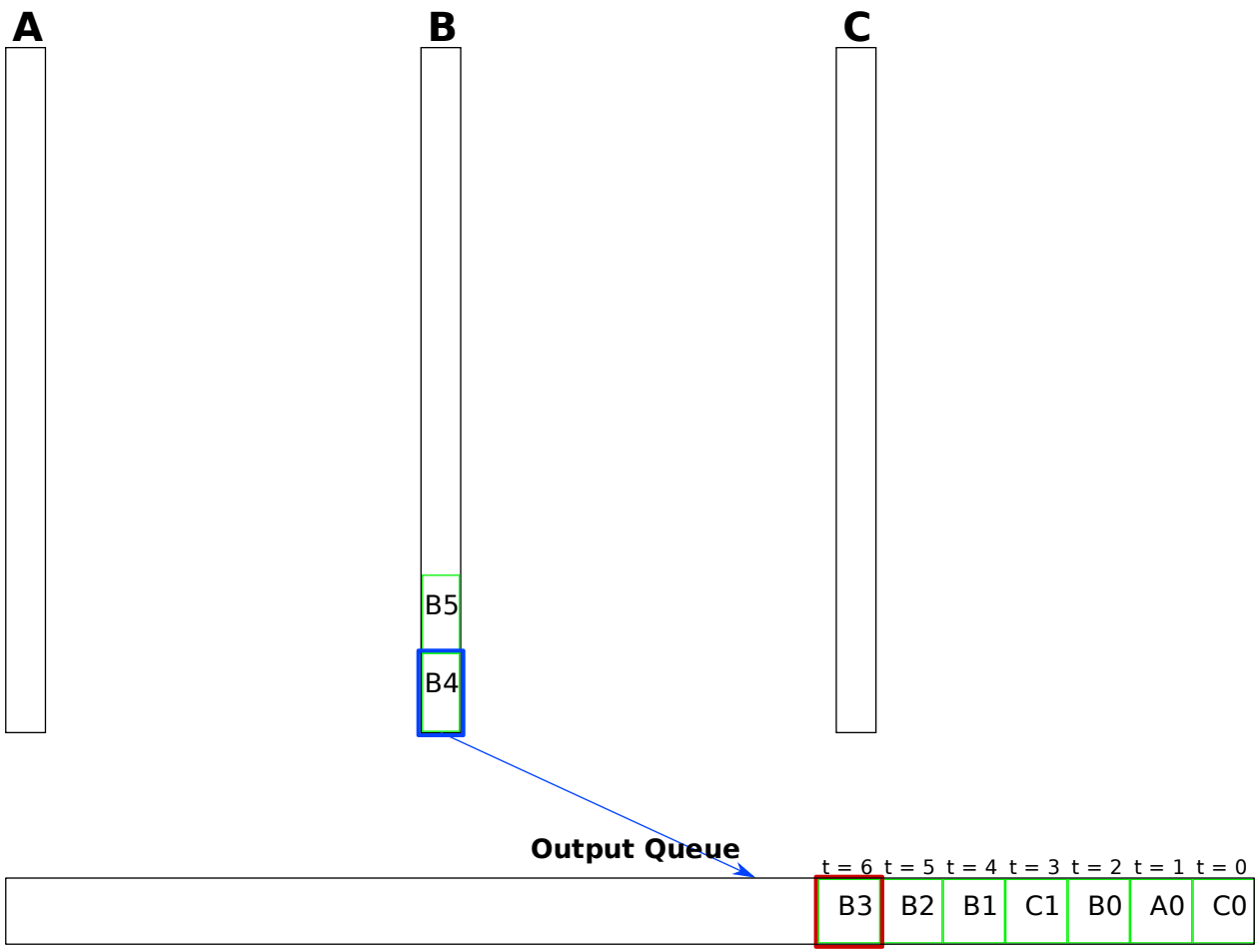
Queue example 5 (3 circuits)



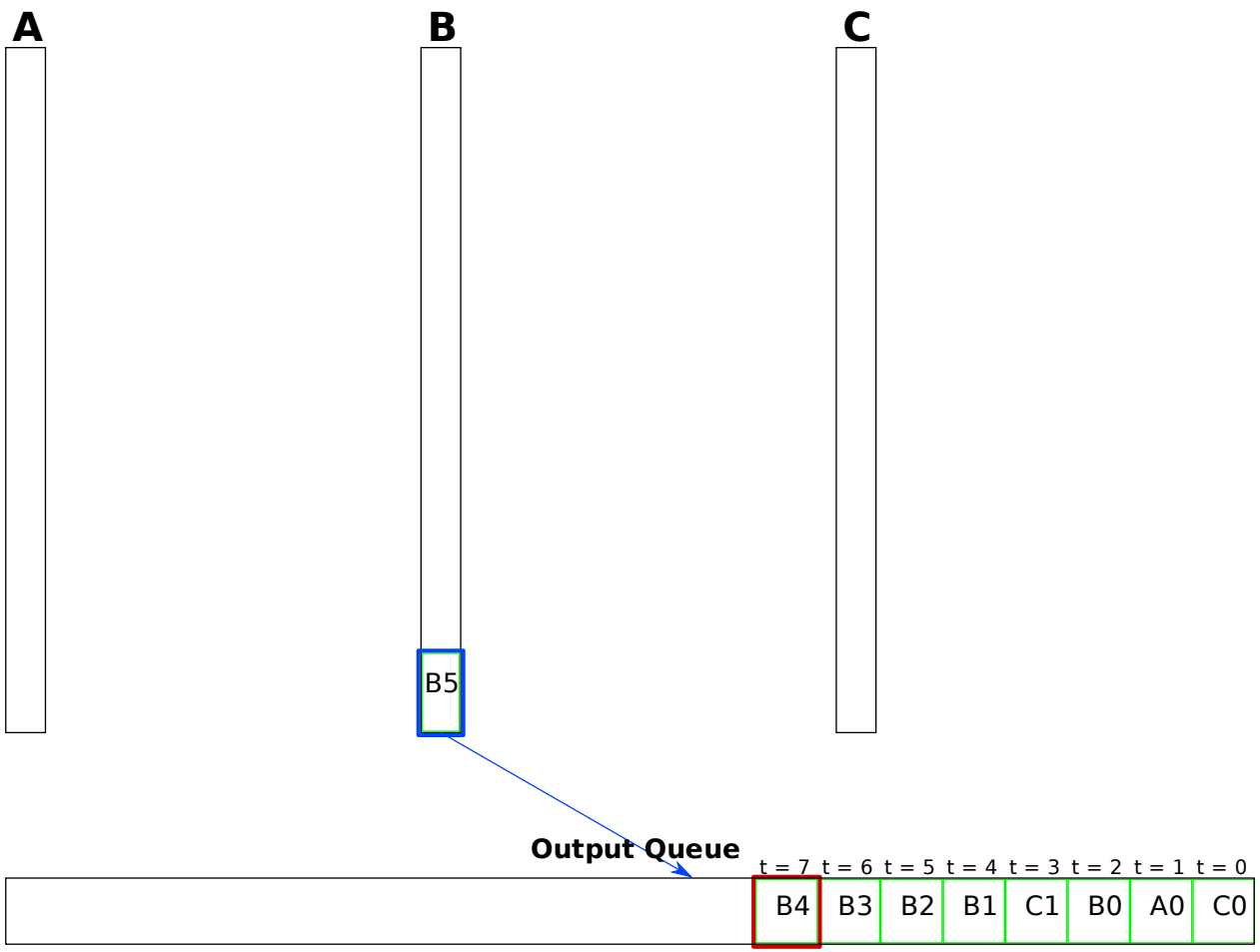
Queue example 6 (3 circuits)



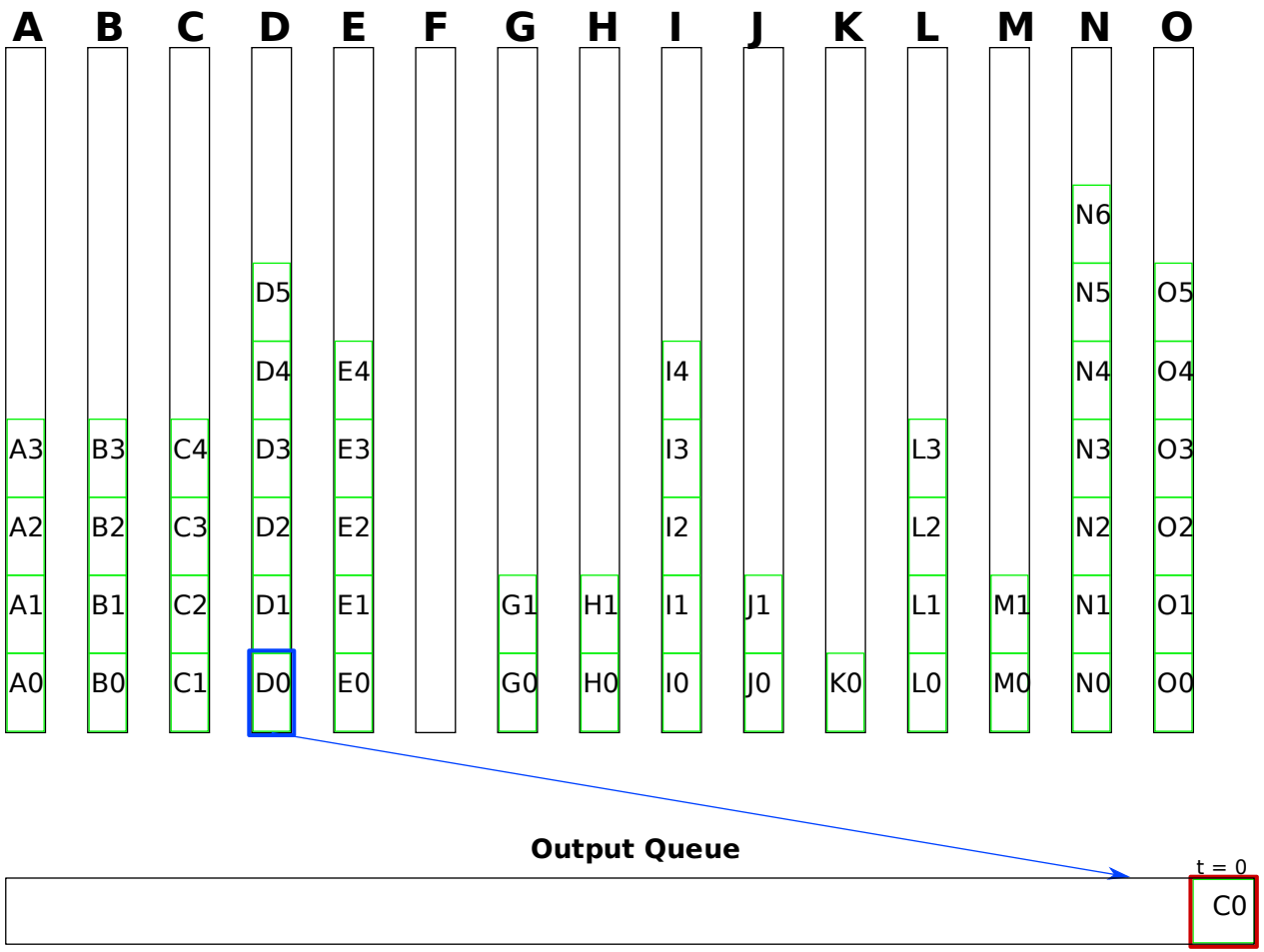
Queue example 7 (3 circuits)



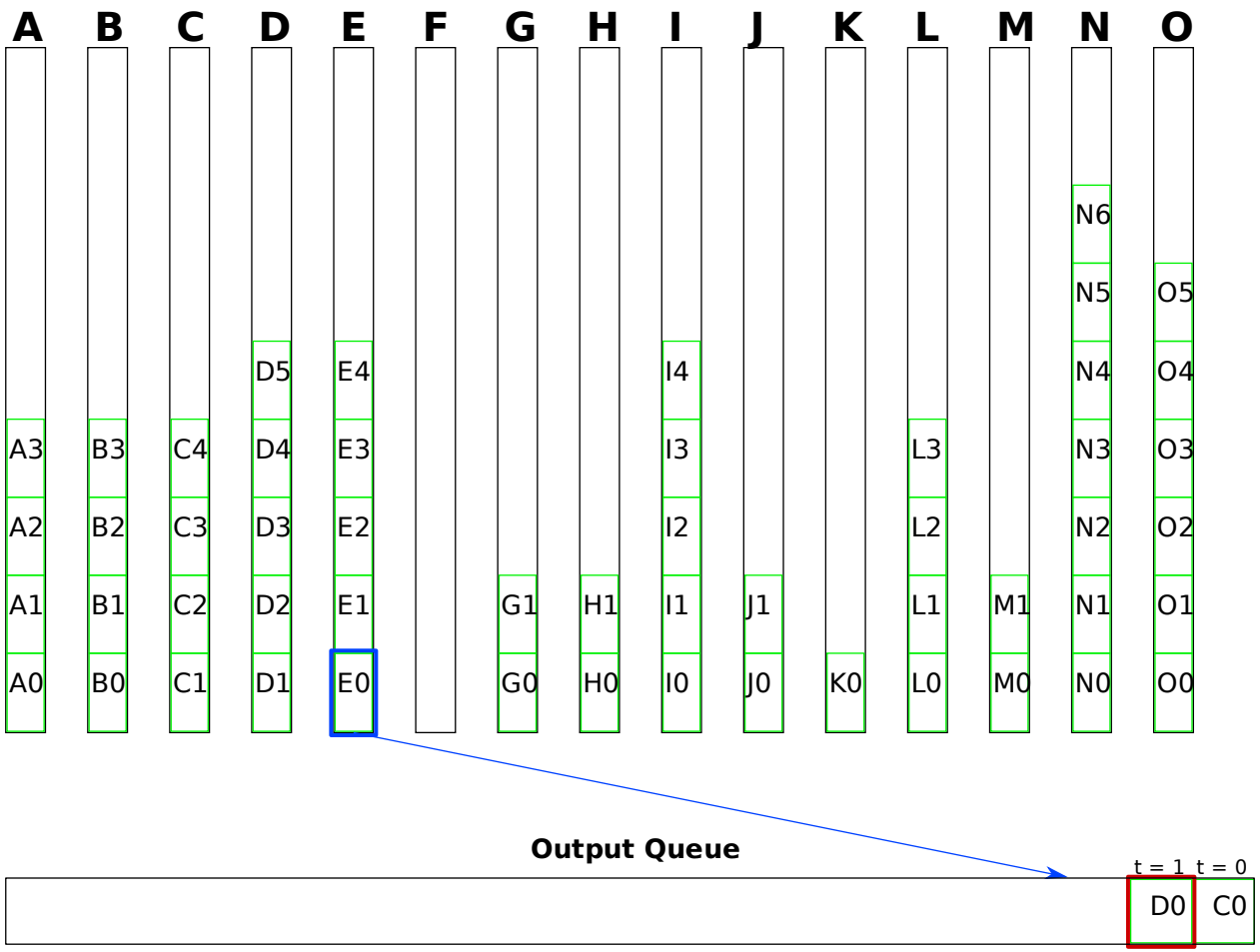
Queue example 8 (3 circuits)



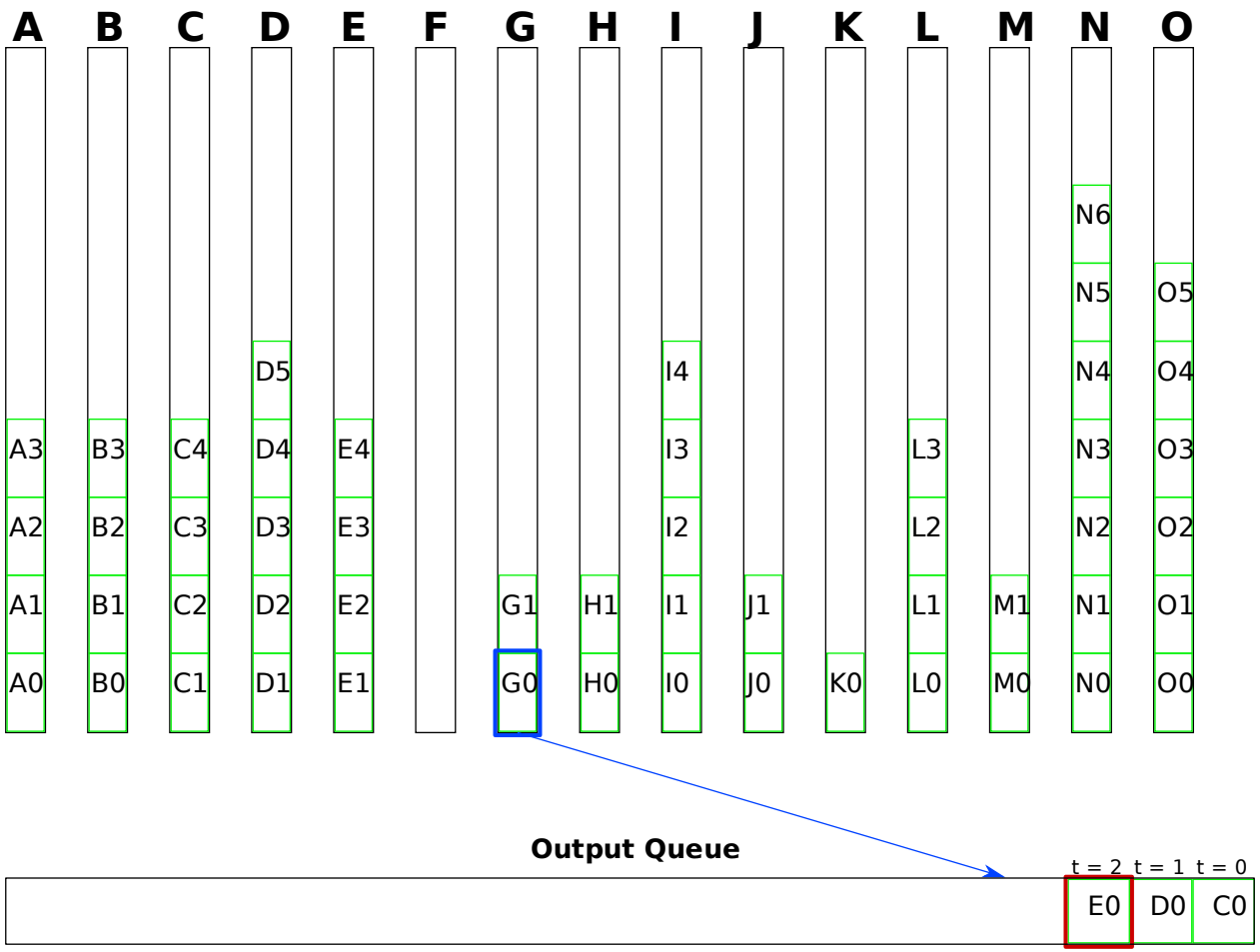
Queue example 1 (15 circuits)



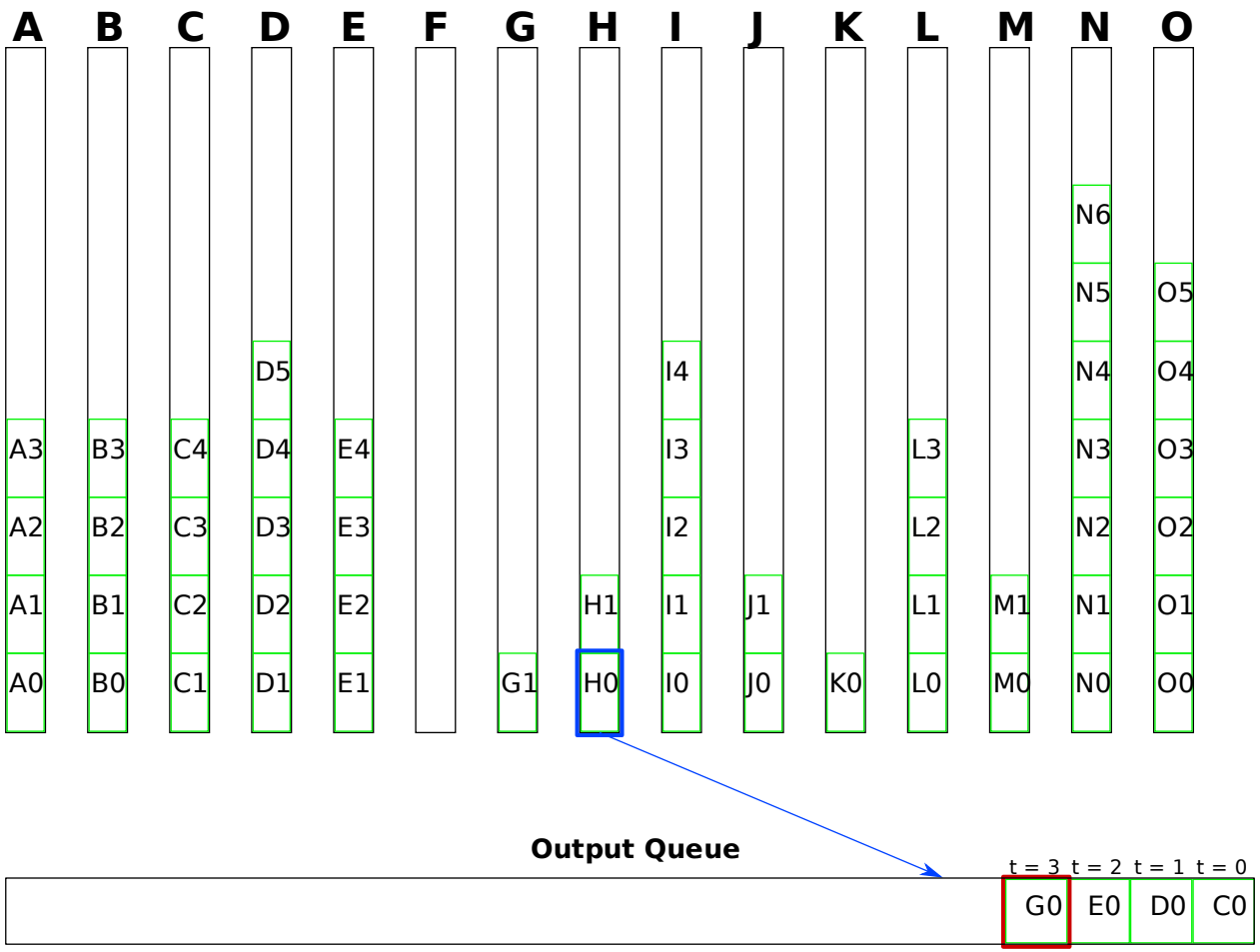
Queue example 2 (15 circuits)



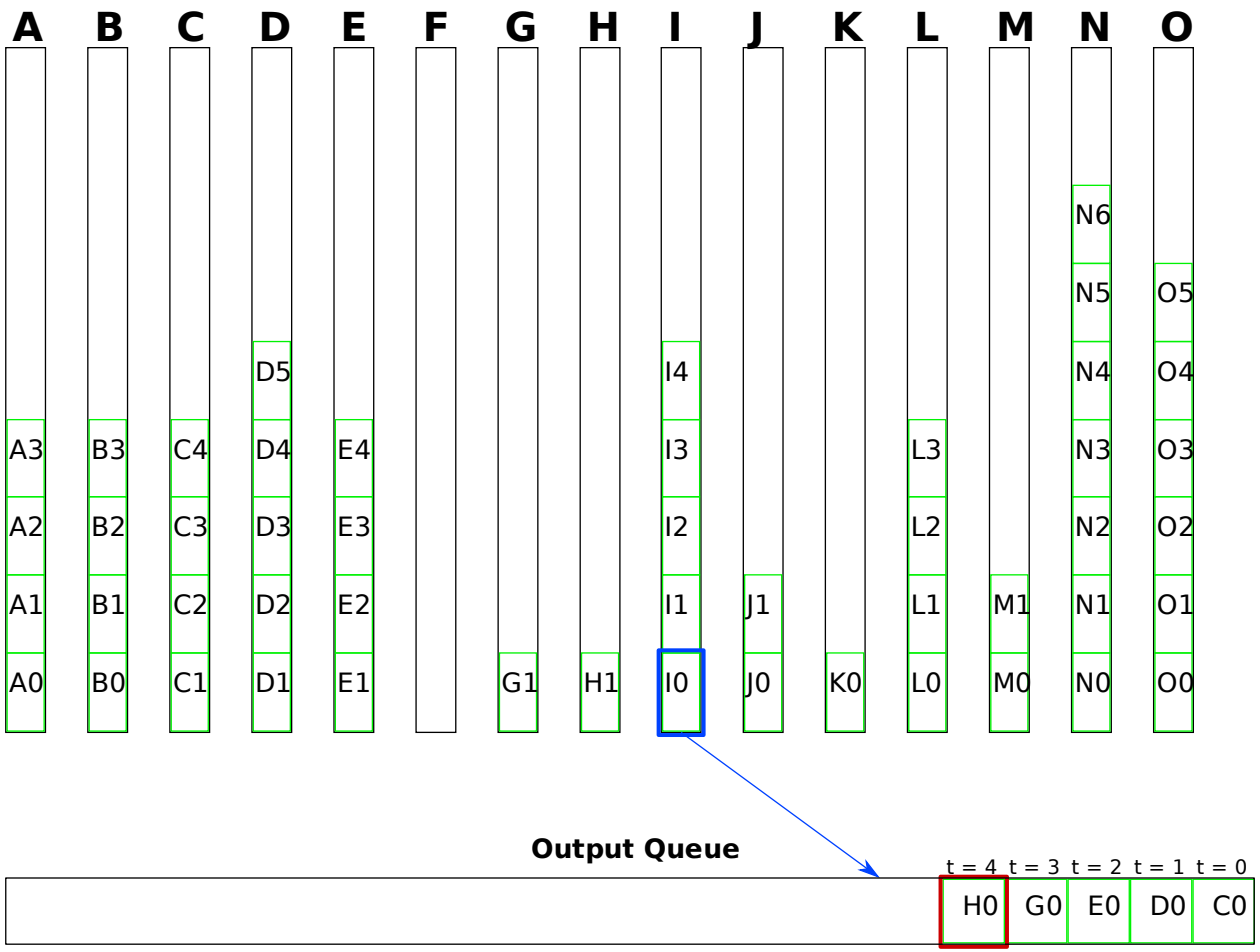
Queue example 3 (15 circuits)



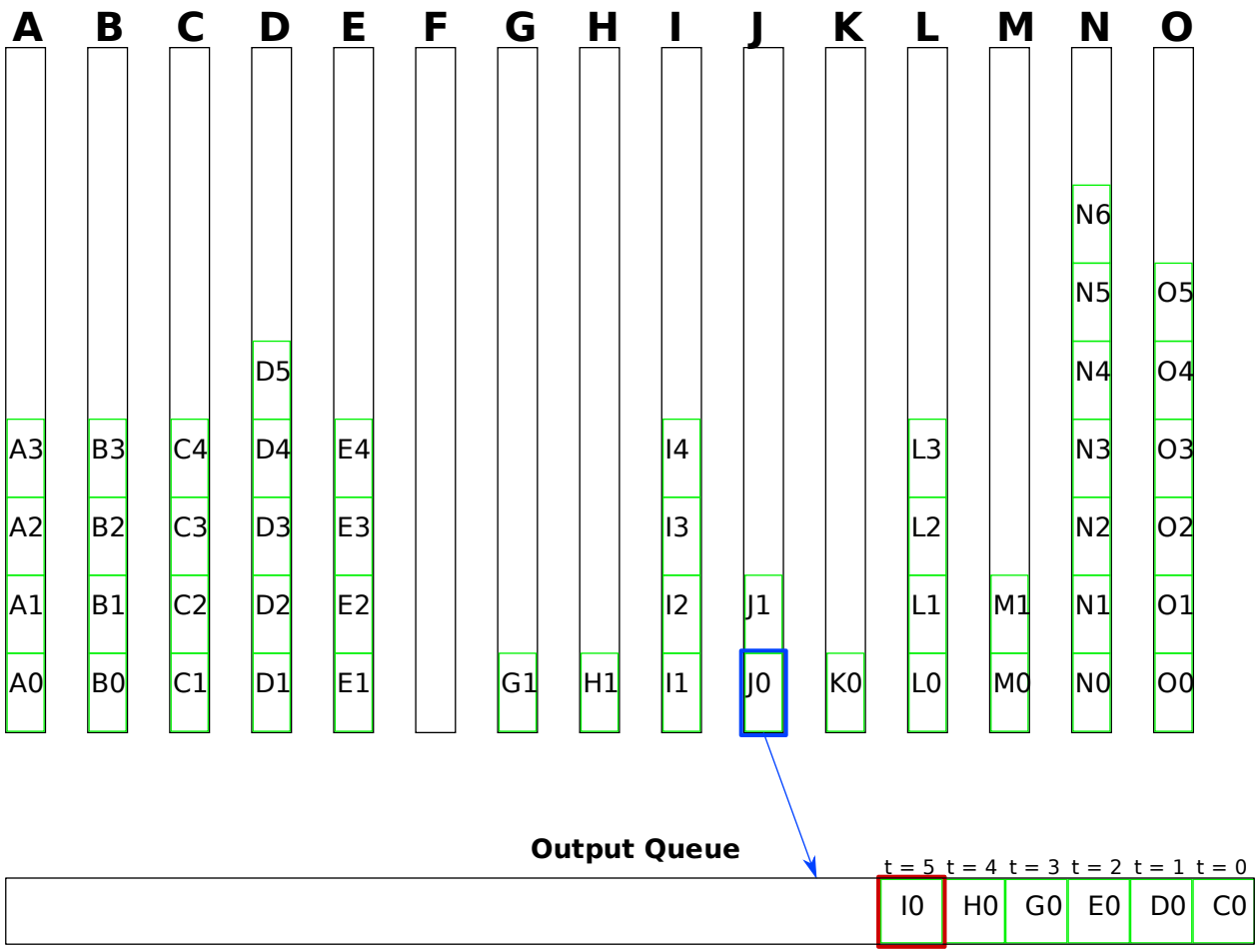
Queue example 4 (15 circuits)



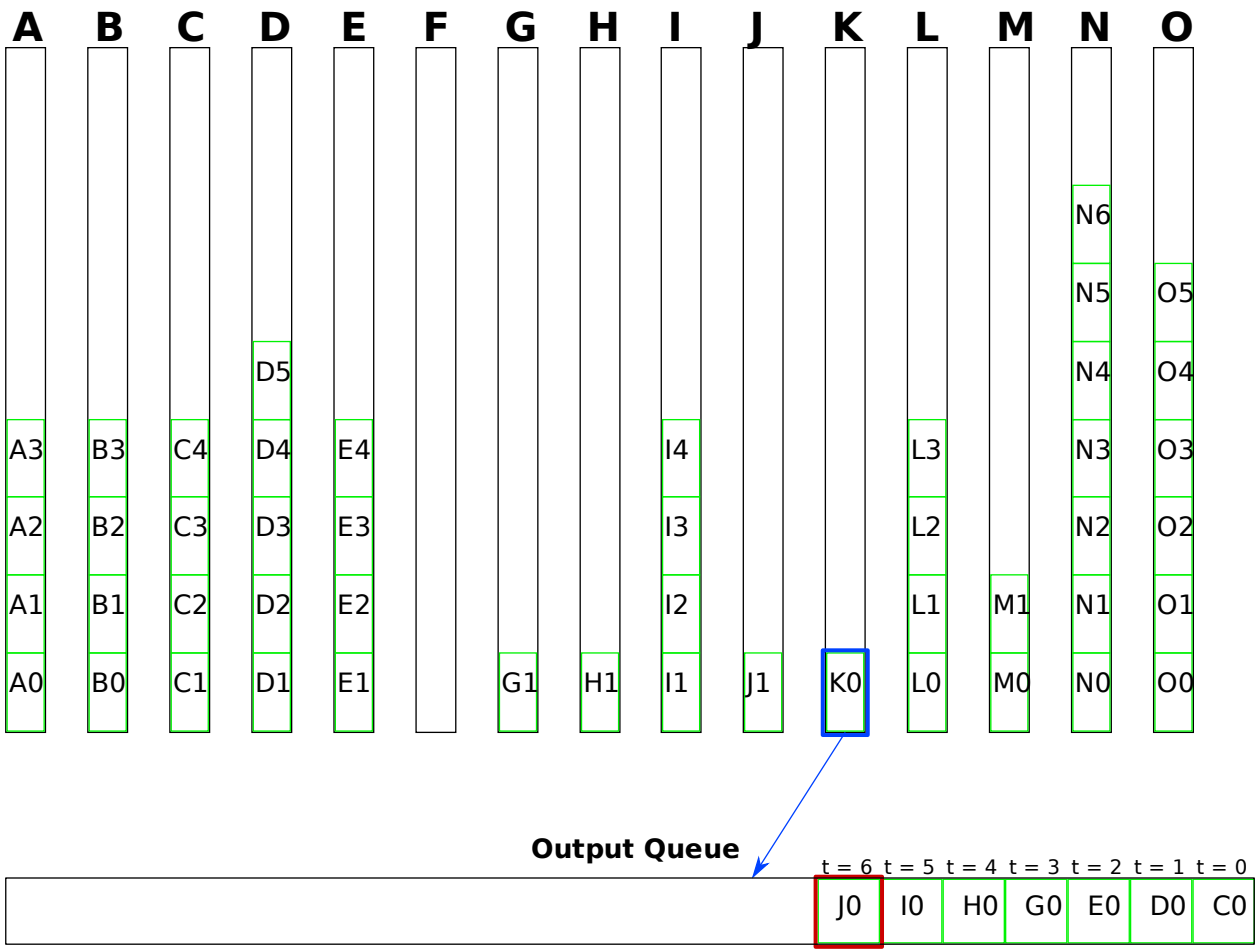
Queue example 5 (15 circuits)



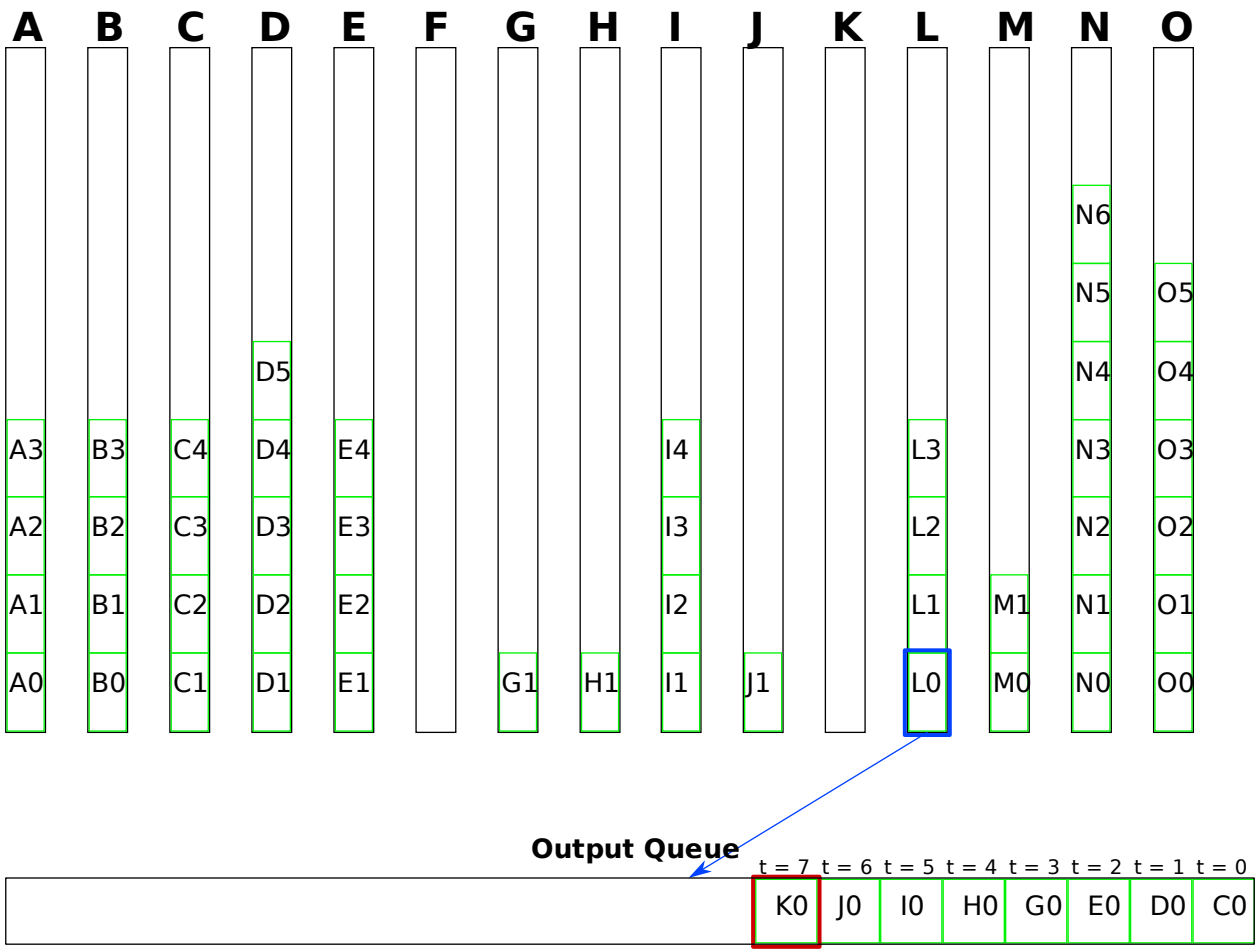
Queue example 6 (15 circuits)



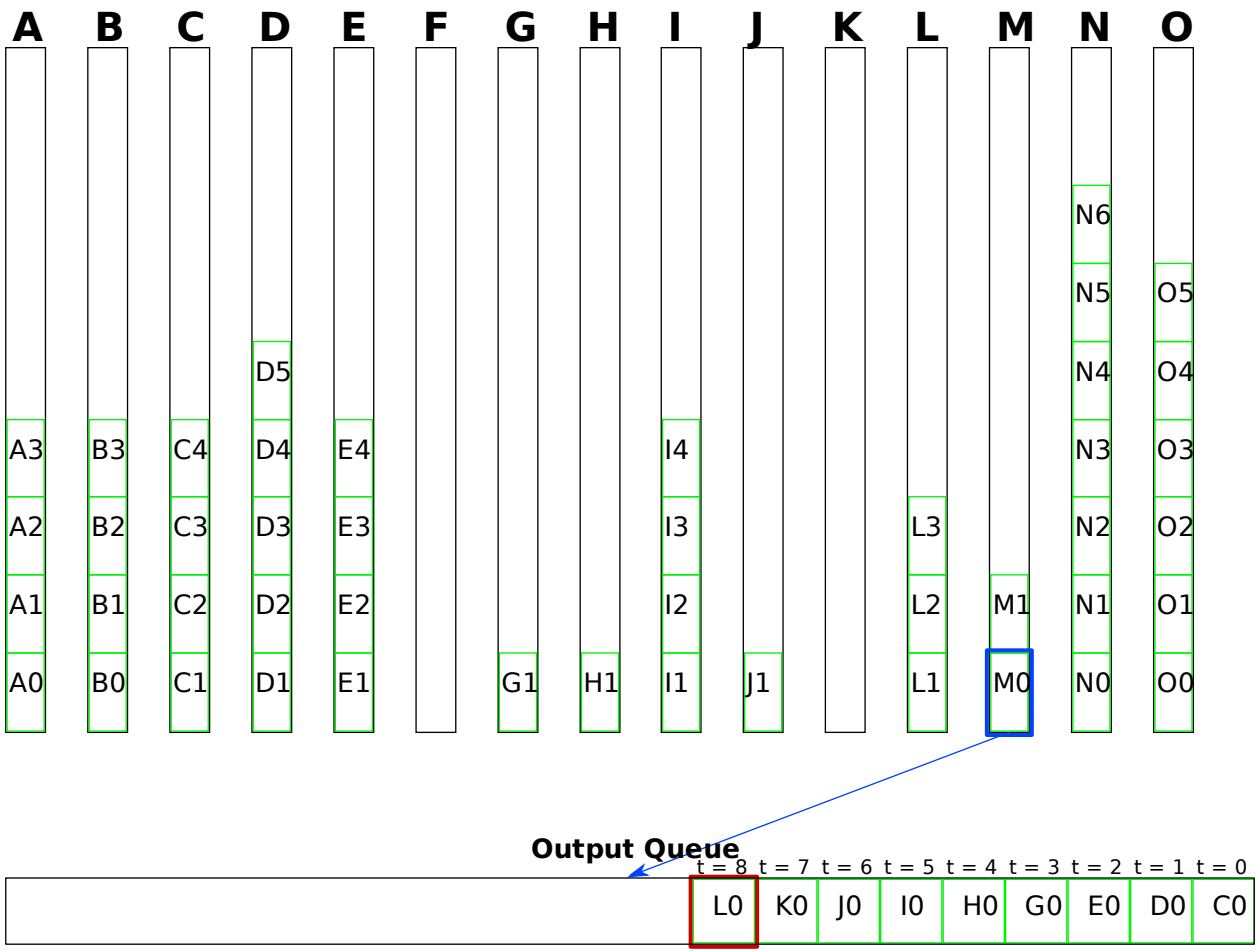
Queue example 7 (15 circuits)



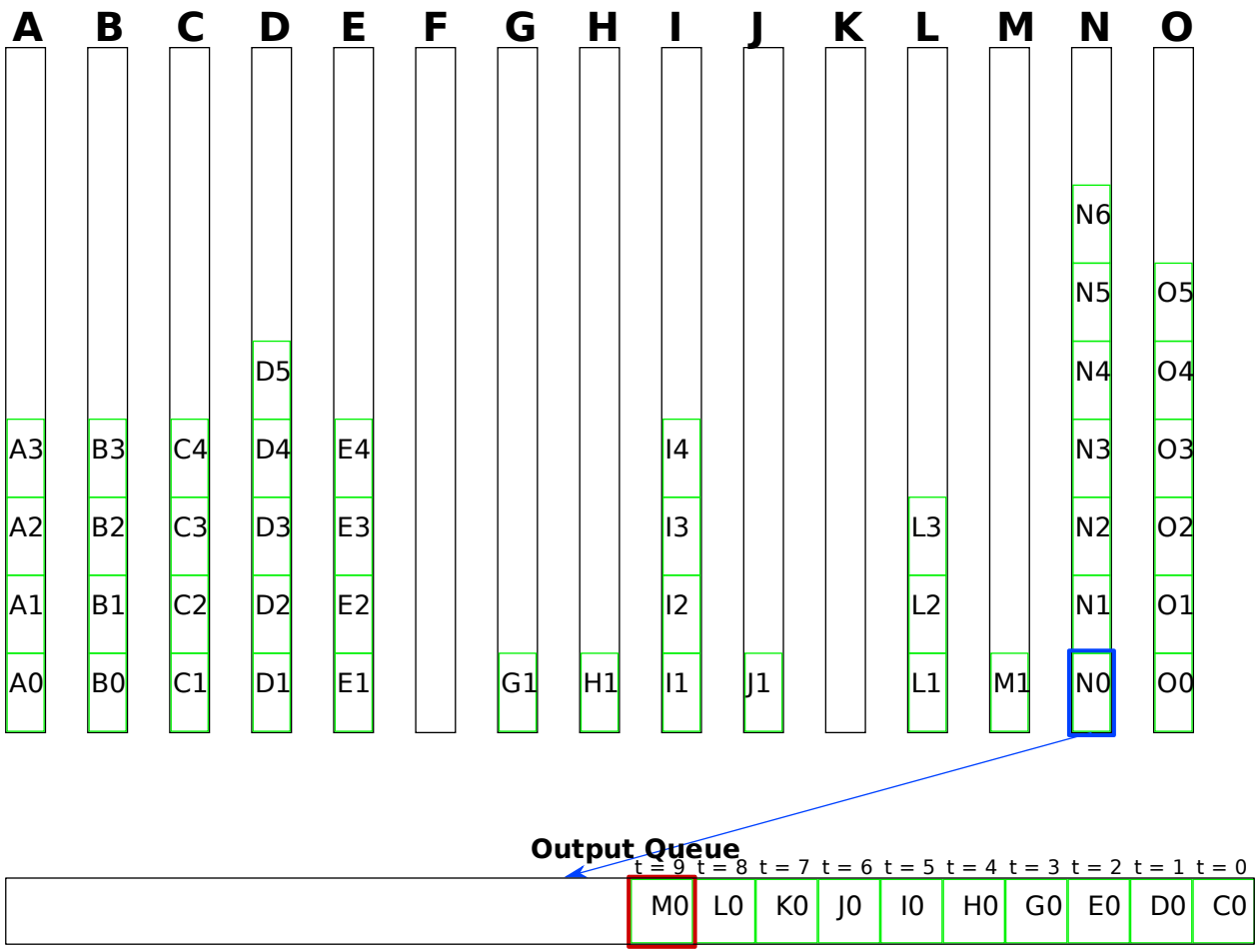
Queue example 8 (15 circuits)



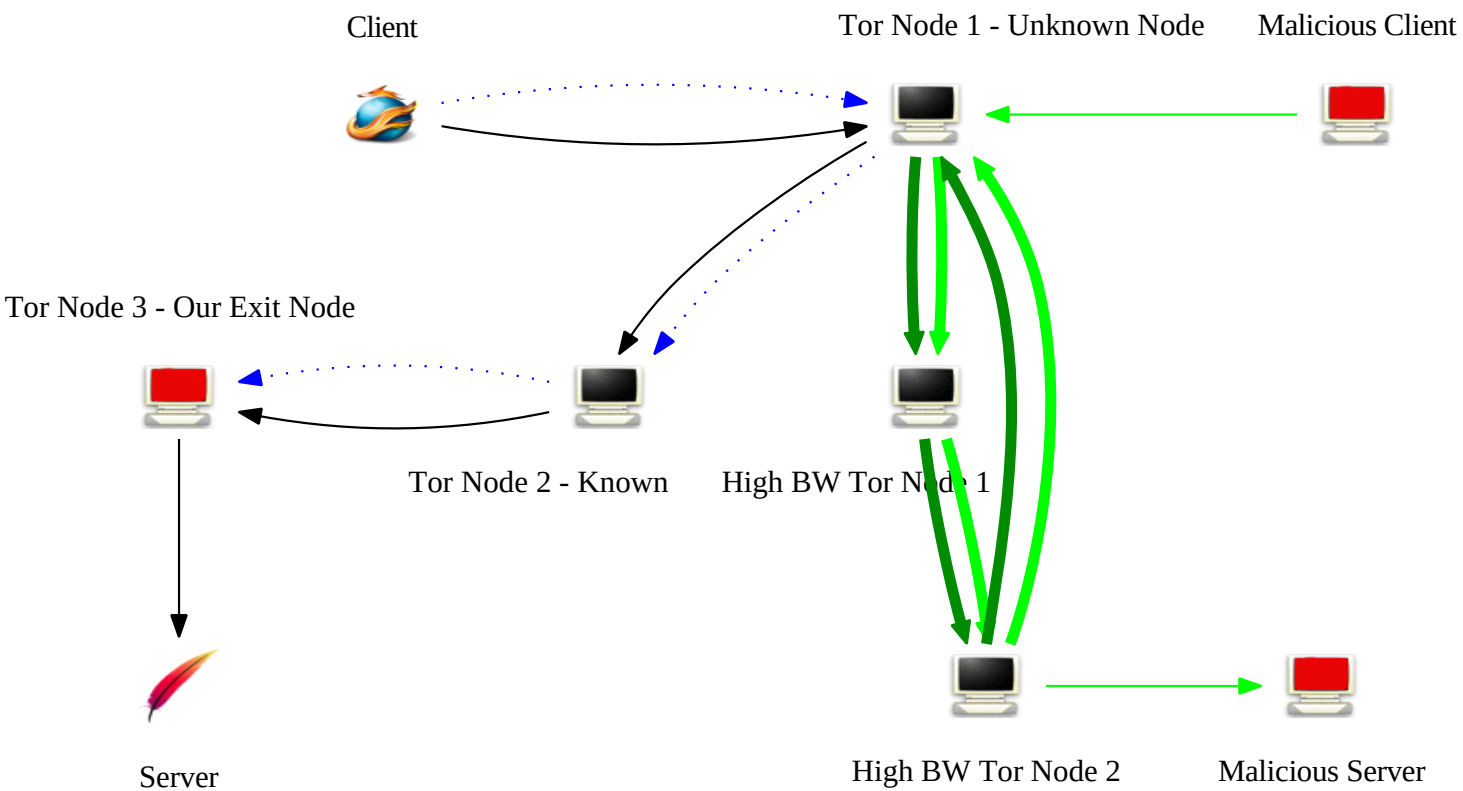
Queue example 9 (15 circuits)



Queue example 10 (15 circuits)



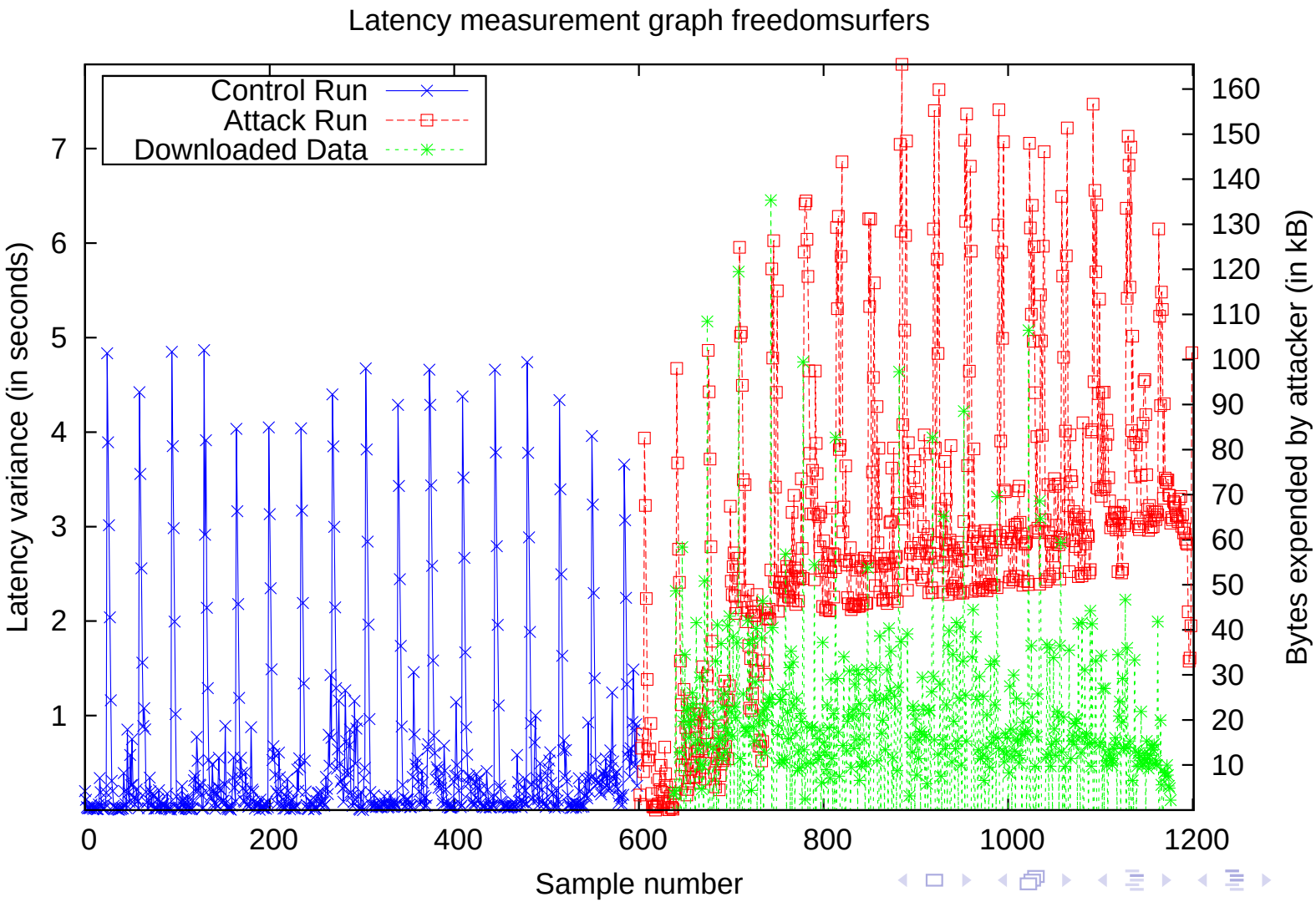
Attack Example



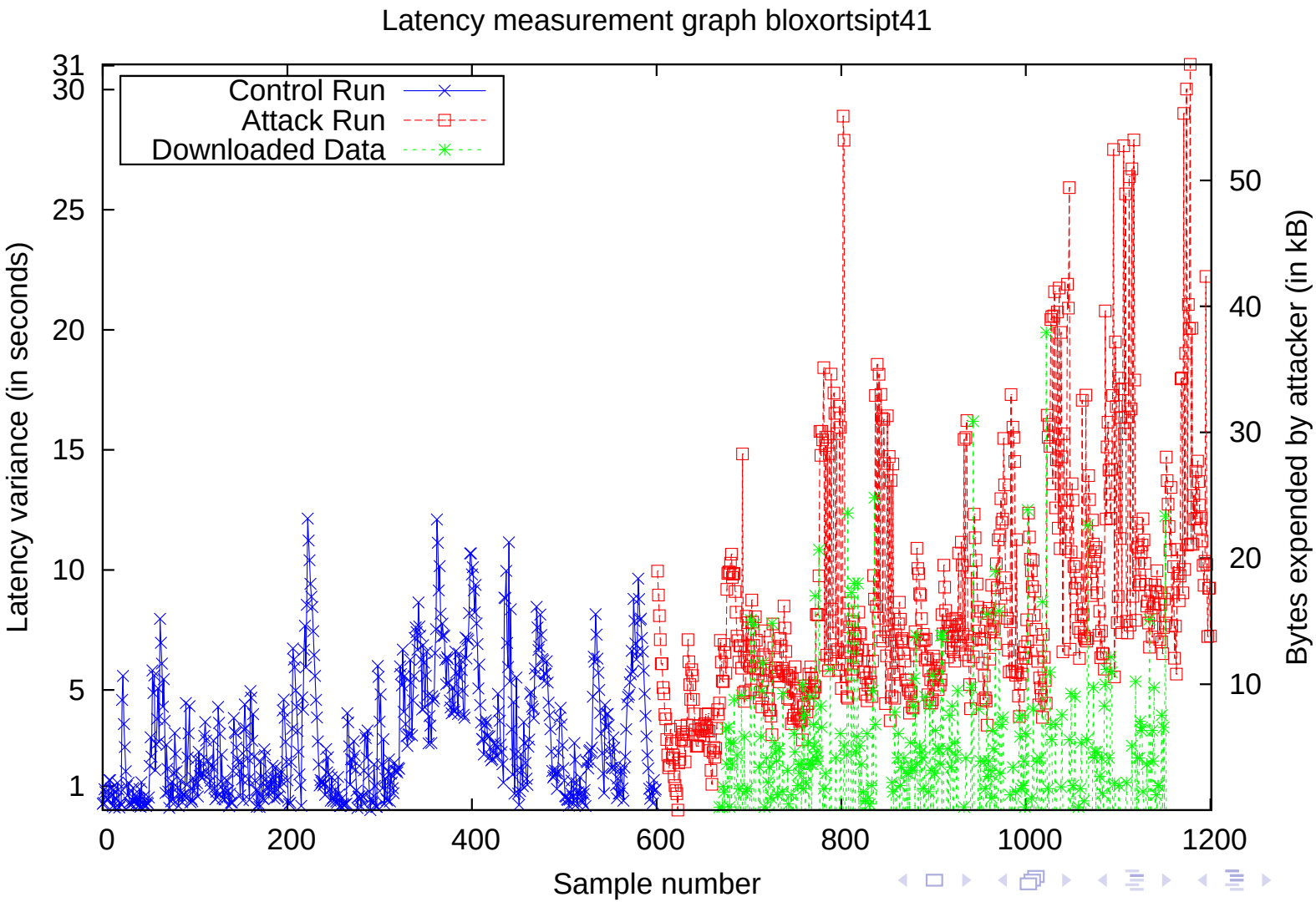
Attack Implementation

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

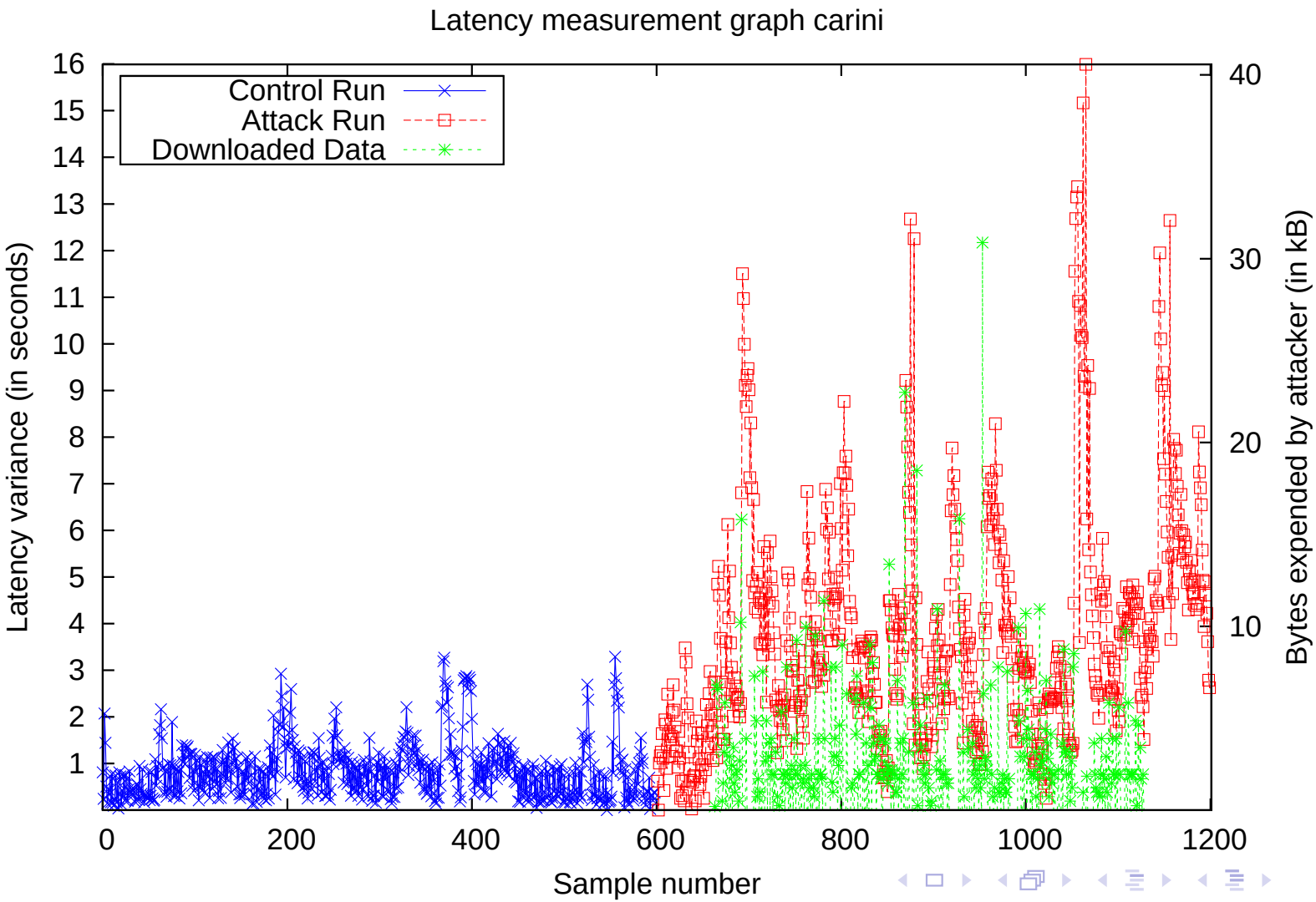
Gathered Data Example (1/8)



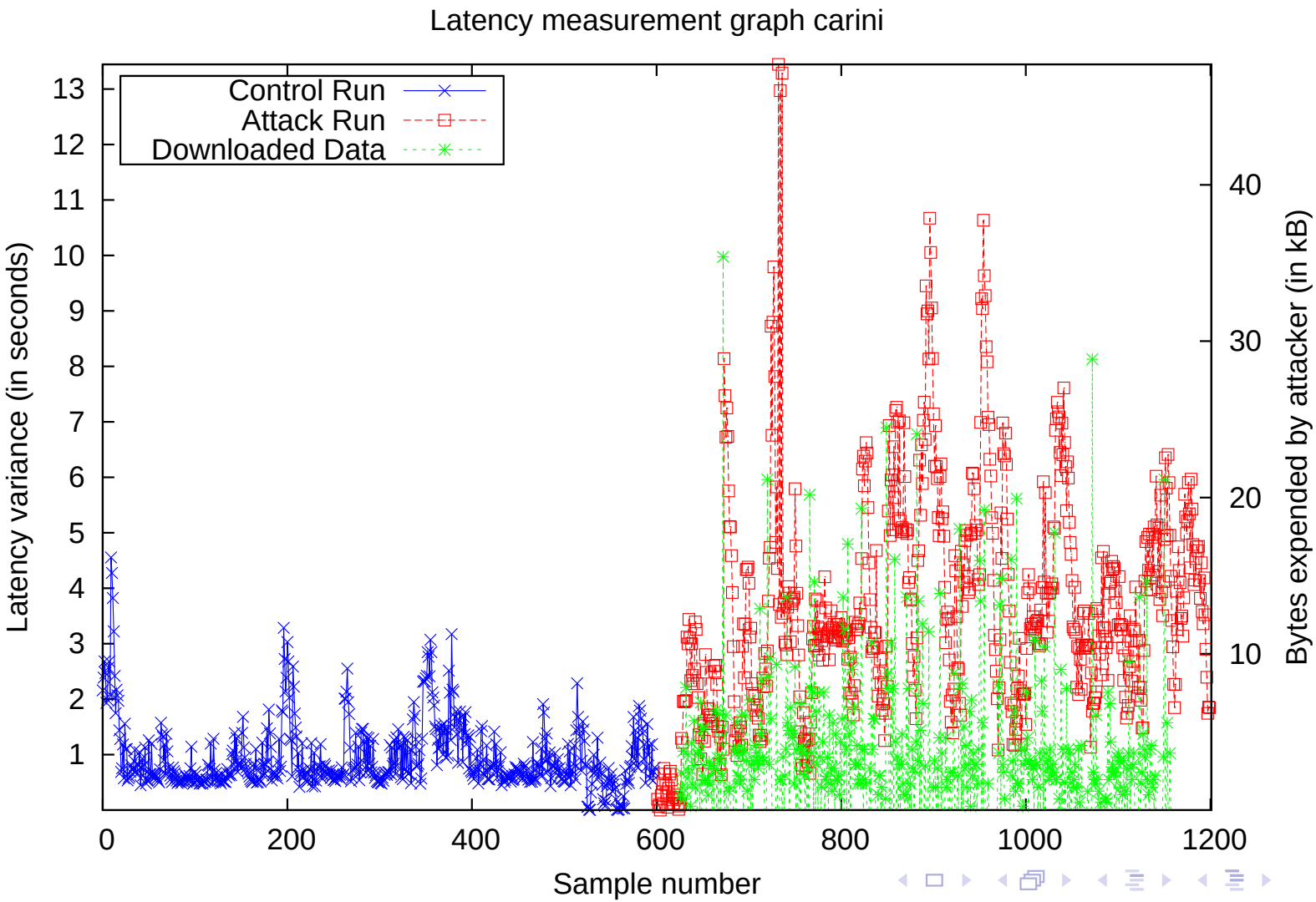
Gathered Data Example (2/8)



Gathered Data Example (3/8)

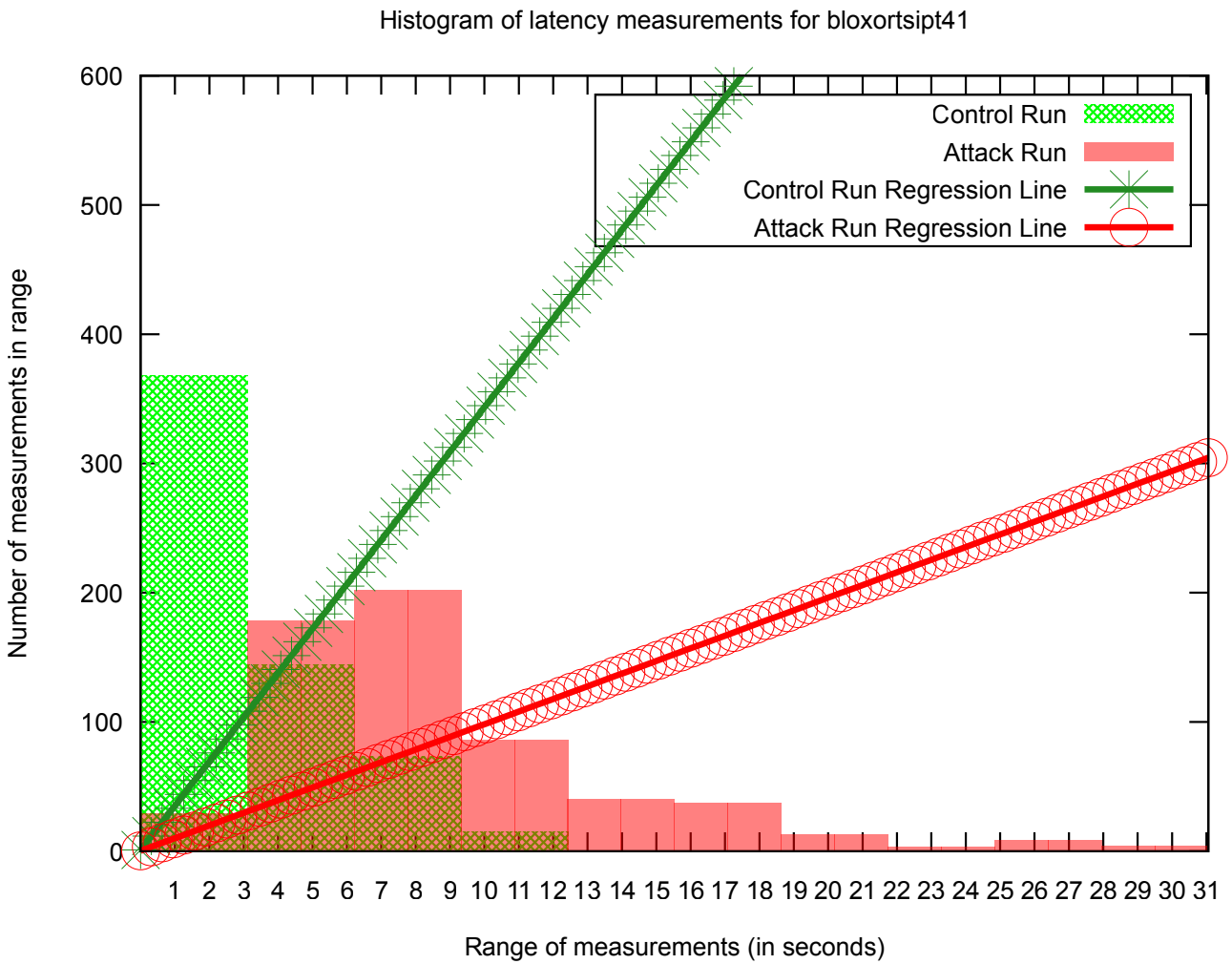


Gathered Data Example (4/8)

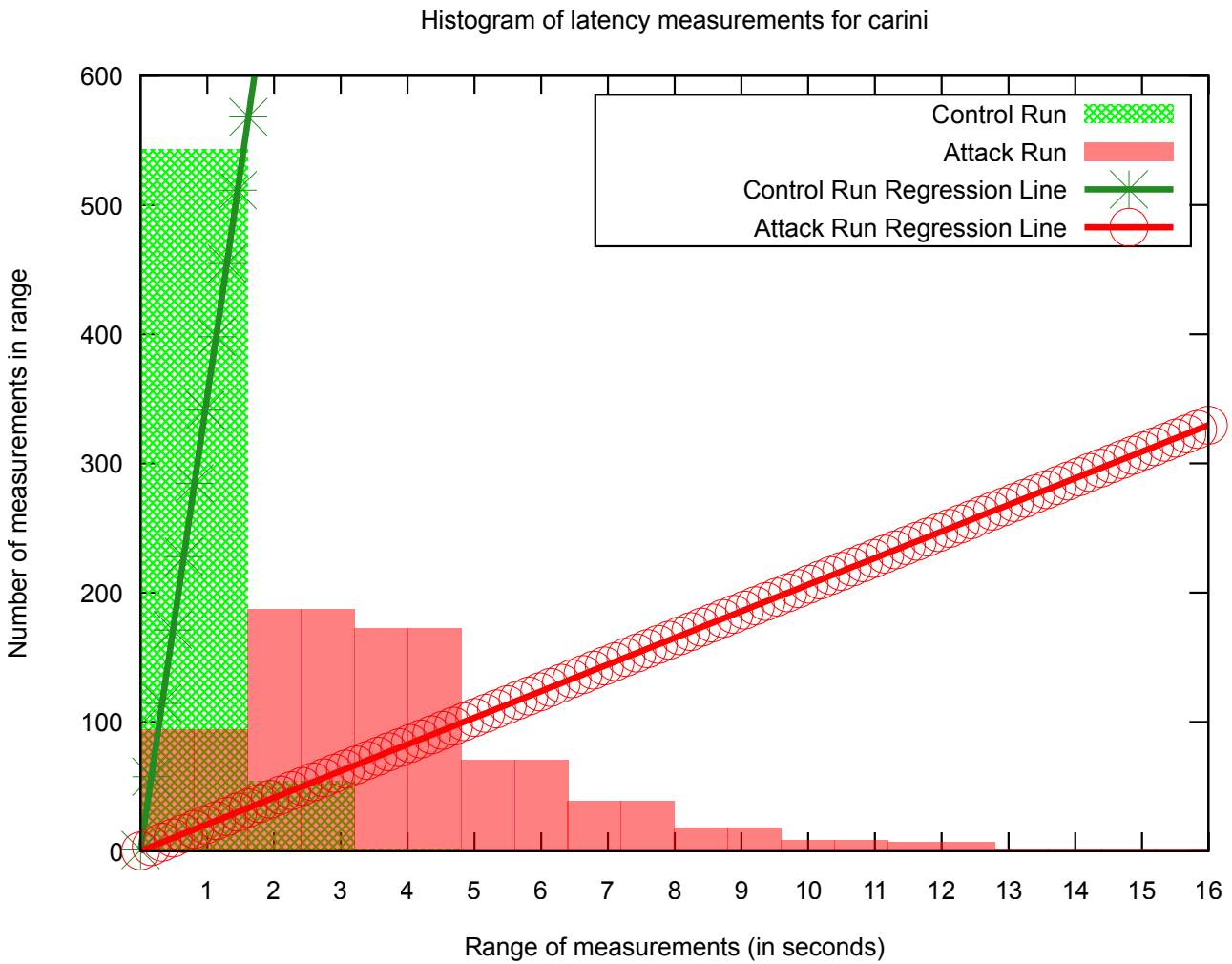


Gathered Data Example (5/8)

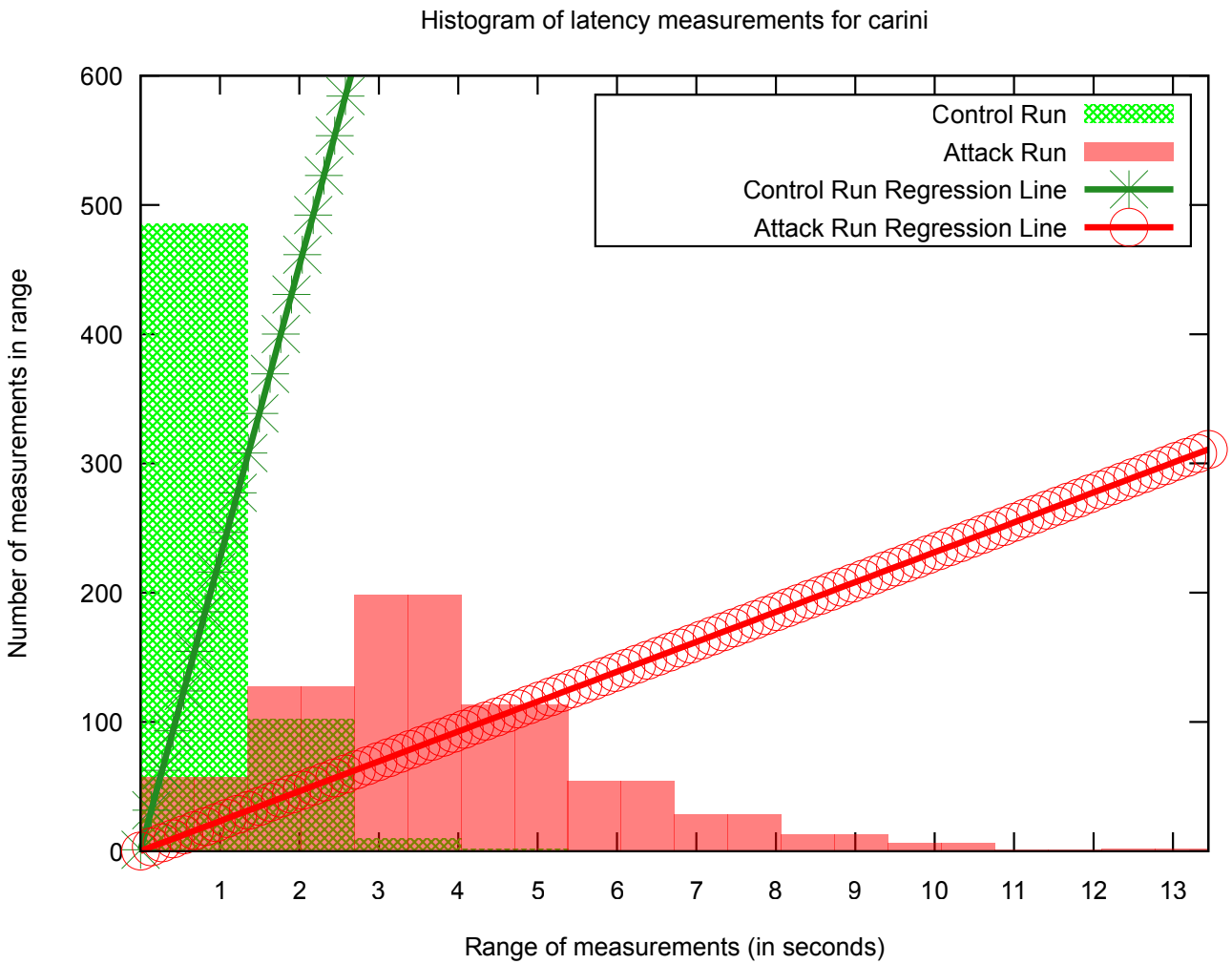
Gathered Data Example (6/8)



Gathered Data Example (7/8)



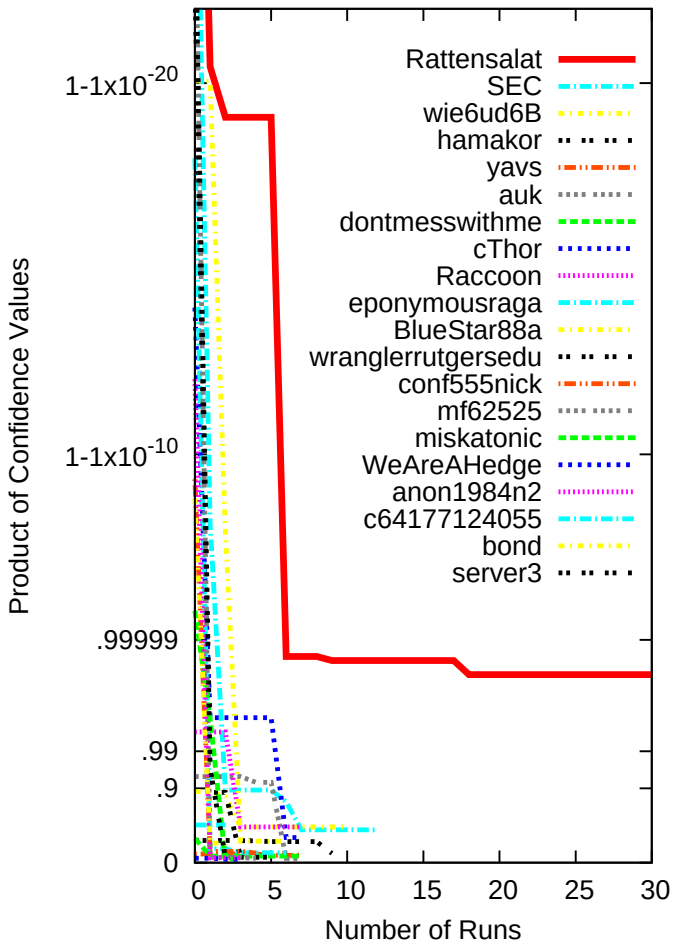
Gathered Data Example (8/8)

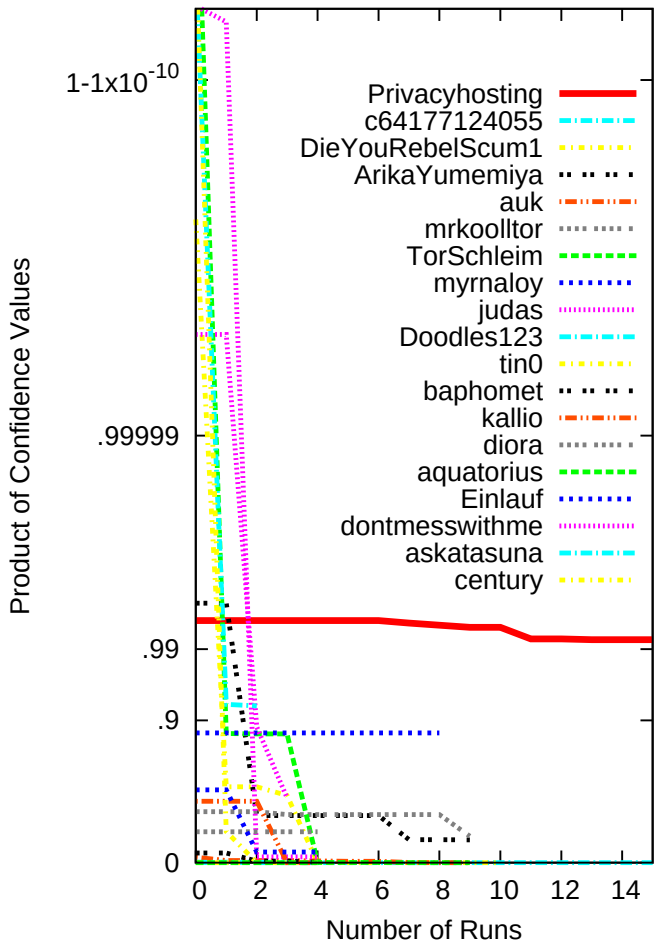


Statistical Analysis

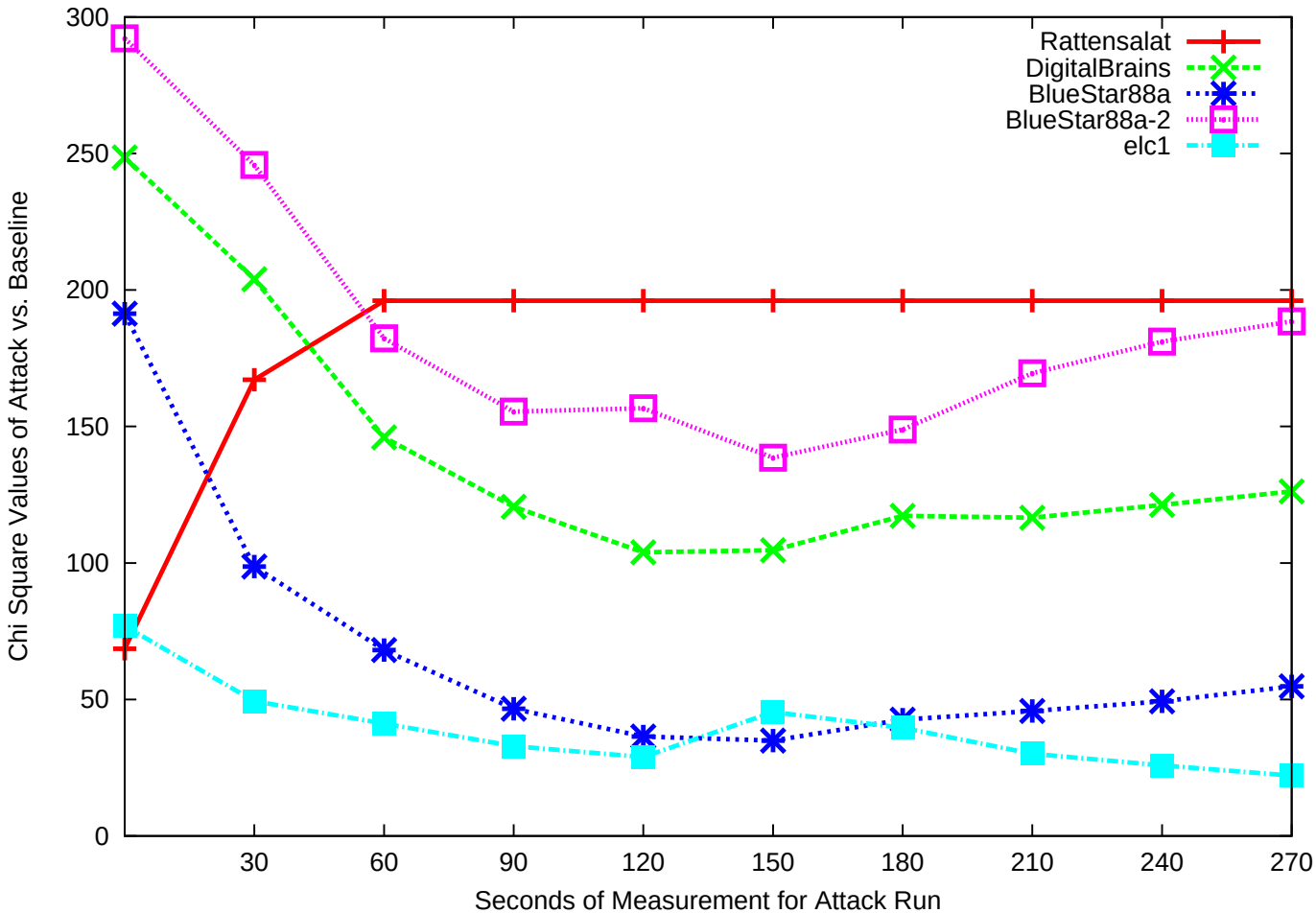
- Use modified χ^2 test
- Compare baseline distribution to attack distribution
- High χ^2 value indicates distribution changed *in the right direction*
- Product of χ^2 confidence values over multiple runs
- Iterate over suspect routers until single node stands out

Cumulative Product of χ^2 p-values





Convergence of χ^2 Values



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
- Attack works on routers with differing bandwidths
- This means that if someone were performing this attack from an exit node, Tor becomes as effective as a network of one-hop proxies

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

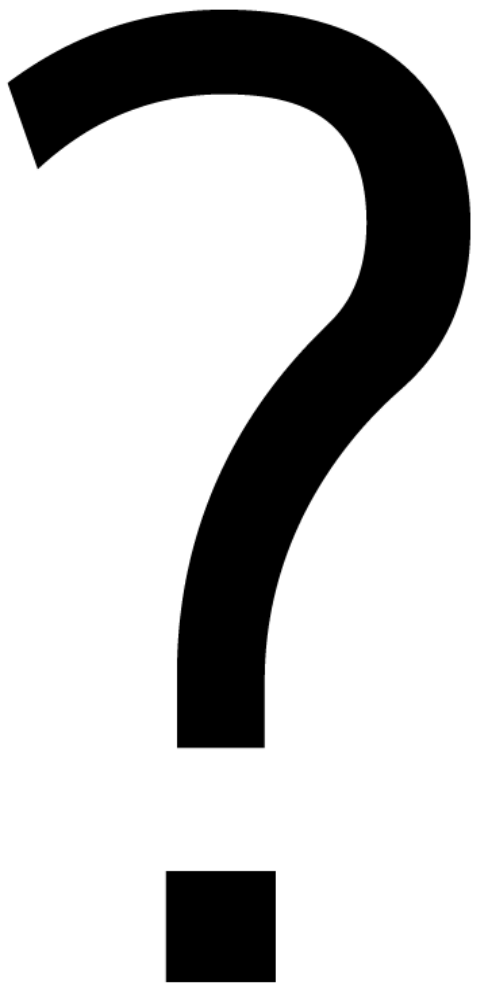
Attack Improvements/Variants

- 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 latency altering attack
- Latency altering attack allows detection of significant changes in latency
- Significant changes in latency reveal paths used

Questions?




R. Dingledine, N. Mathewson, and P. Syverson.
Tor: The second-generation onion router.
In *Proceedings of the 13th USENIX Security Symposium*, August 2004.


N. S. Evans, R. Dingledine, and C. Grothoff.
A practical congestion attack on tor using long paths.
In *18th USENIX Security Symposium*, pages 33–50. USENIX, 2009.