Enabling Refinable Cross-Host Attack Investigation with Efficient Data Flow Tagging and Tracking

Yang Ji, Sangho Lee, Mattia Fazzini, Joey Allen, Evan Downing,
Taesoo Kim, Alessandro Orso, and Wenke Lee

USENIX Security Symposium 2018

August 17, 2018

Georgia Tech

Advanced attacks involve multiple hosts

Taiwan ATM heist linked to European hacking spree: security firm

Through a series of systematic, lateral movements (see illustration, below), they ultimately stole money from ATMs, where criminal associates would retrieve the cash.

Goal of investigation:
accurate, efficient, supporting multi-hosts

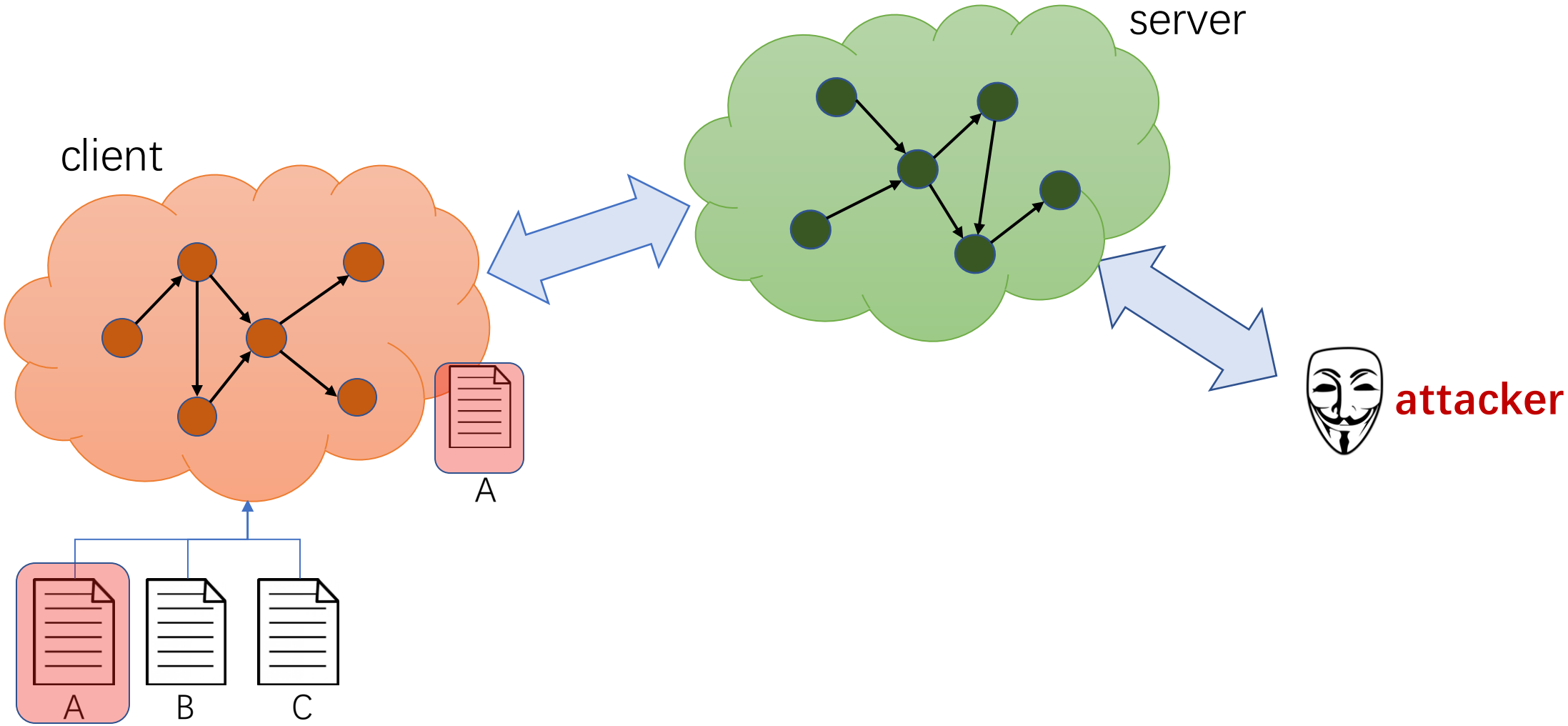
served Carbanak actors employing a handful of unique Trojans, along with freely available malware, to persist and move laterally once a network foothold was established. While

GitPwnd

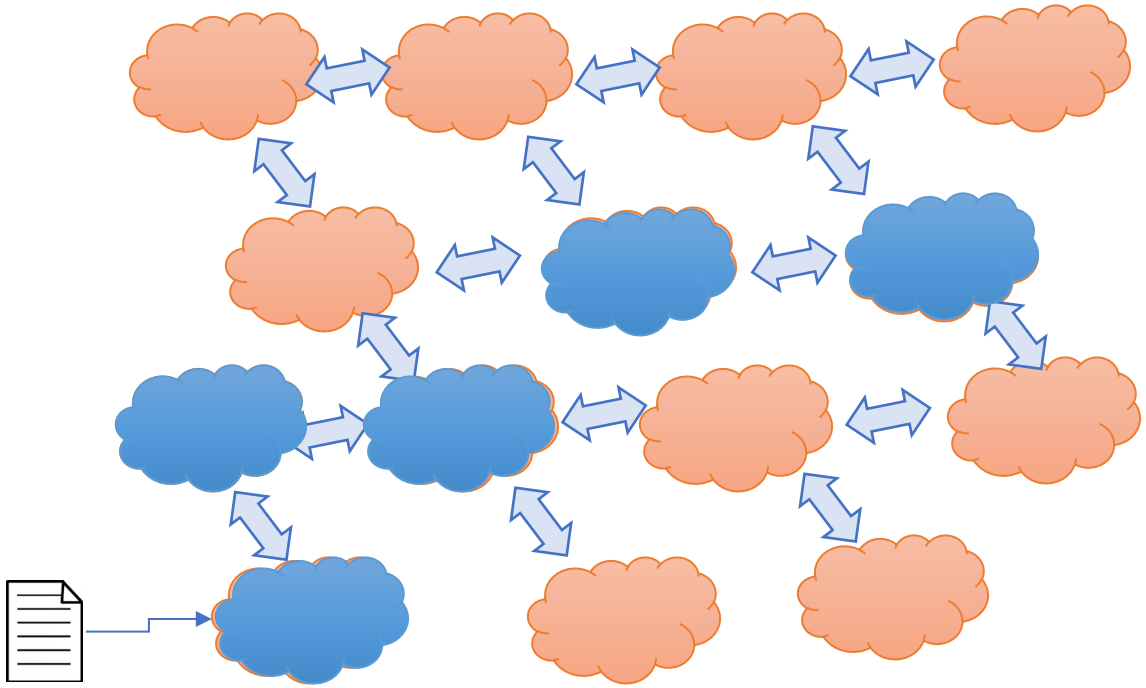
GitPwnd is a tool to aid in network penetration tests. GitPwnd allows an attacker to send commands to compromised machines and receive the results back using a git repo as the command and control transport layer. By using git as the communication mechanism, the compromised machines don't need to communicate directly with your attack server that is likely at a host or IP that's untrusted by the compromised machine.

Answer queries:

- What data were leaked to the attacker?
- Was B leaked?
- How was A leaked step in step?



Distributed setting, e.g., P2P network



Analyzing data flow across hosts is hard

- False positive dependencies
- Data dependencies across multiple hosts
- Amplified analysis cost

Resolving false positive dependencies

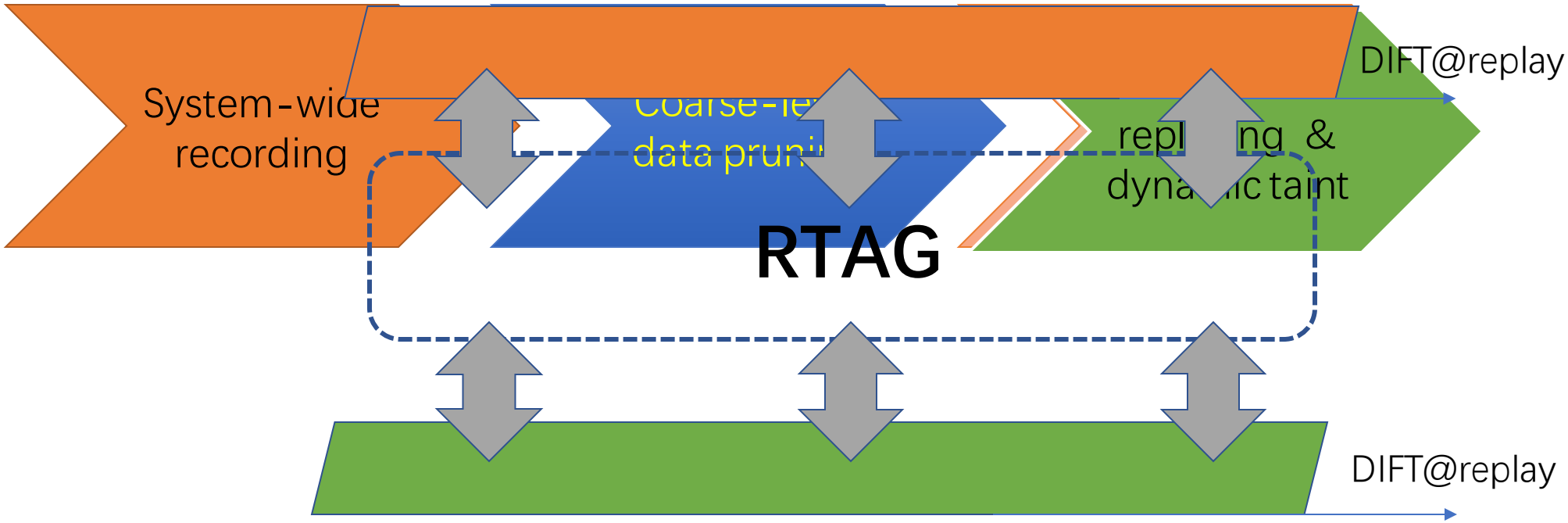
- Using dynamic taint analysis at runtime
  - Suffering from high overhead
  - Cloudfence (RAID ‘13), TaintExchange (IWSEC ‘12)
- Refinable attack investigation (**We take this direction**)
  - Record replay + dynamic taint analysis
  - Arnold (OSDI '14), RAIN (CCS '17)

Analyzing data flow across hosts is hard

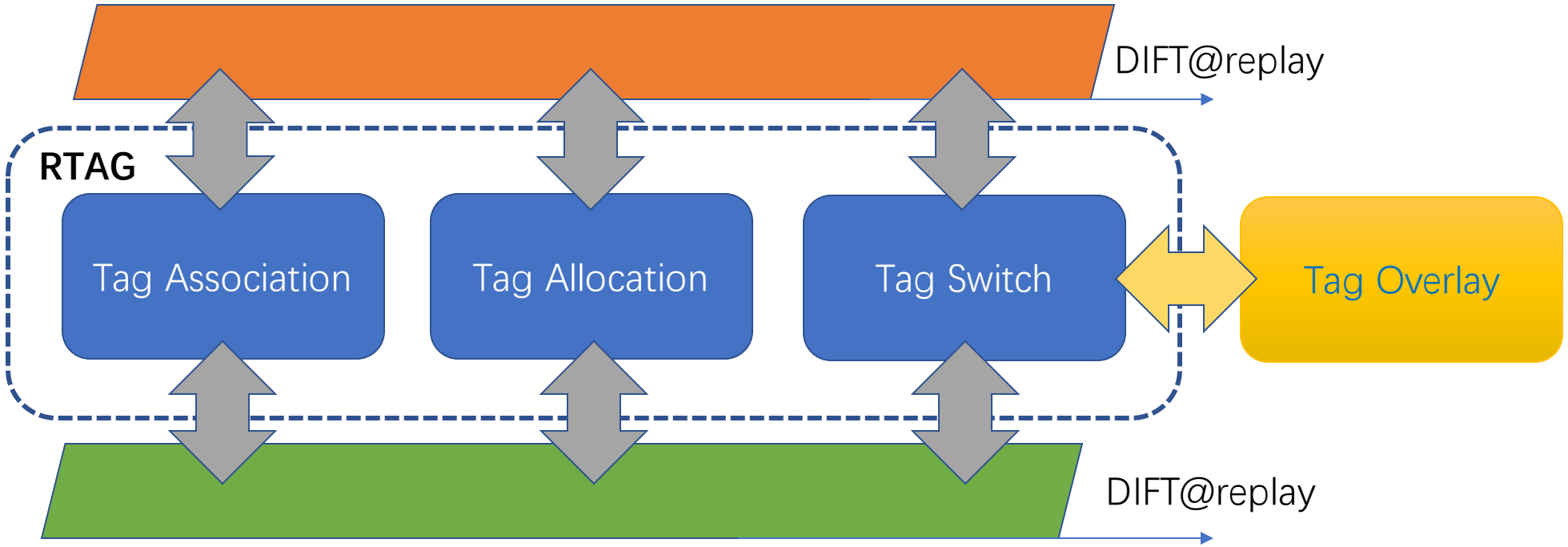
- False positive dependencies
  - **Record replay + dynamic information flow tracking (DIFT)**
- Data dependencies across multiple hosts
  - **Enable tag-dependency-free, independent and parallel replays**
- Amplified analysis cost
  - **Optimize the analysis time and memory cost**

Our approach

- Enable ***independent*** and ***parallel*** replayed DIFT
- Reduce the memory cost of DIFT by optimally allocating tag size for each DIFT task



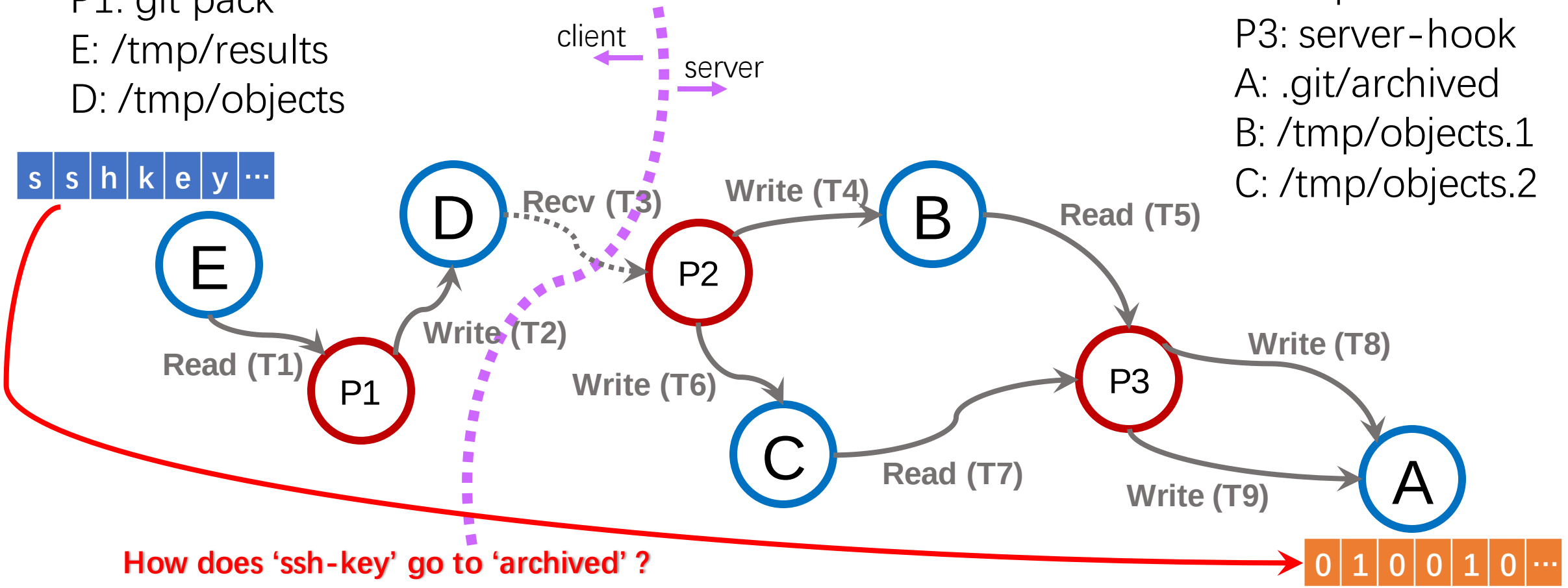
Overview


DIFT@replay
RTAG
Tag Association
Tag Allocation
Tag Switch
Tag Overlay
DIFT@replay

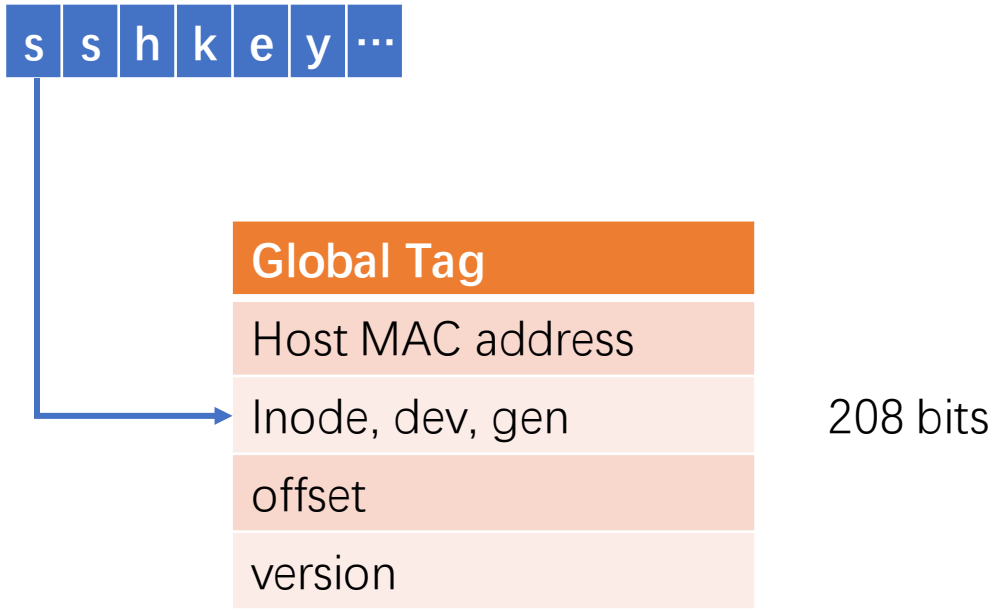
Gitpwnd data exfiltration

@ 10.0.0.1:
P1: git pack
E: /tmp/results
D: /tmp/objects

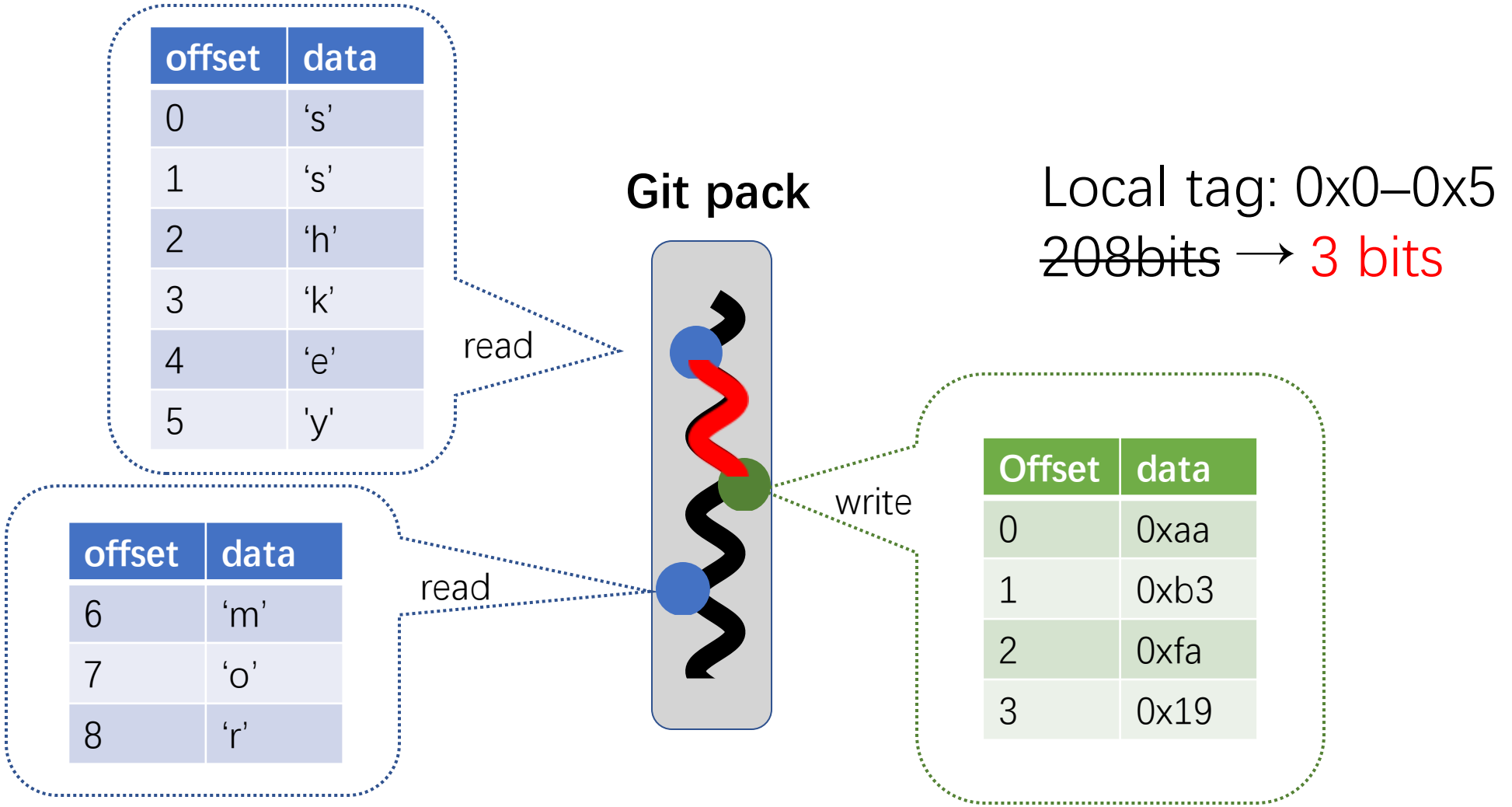
@ 10.0.0.2:
P2: scp
P3: server-hook
A: .git/archived
B: /tmp/objects.1
C: /tmp/objects.2



Length of global tag



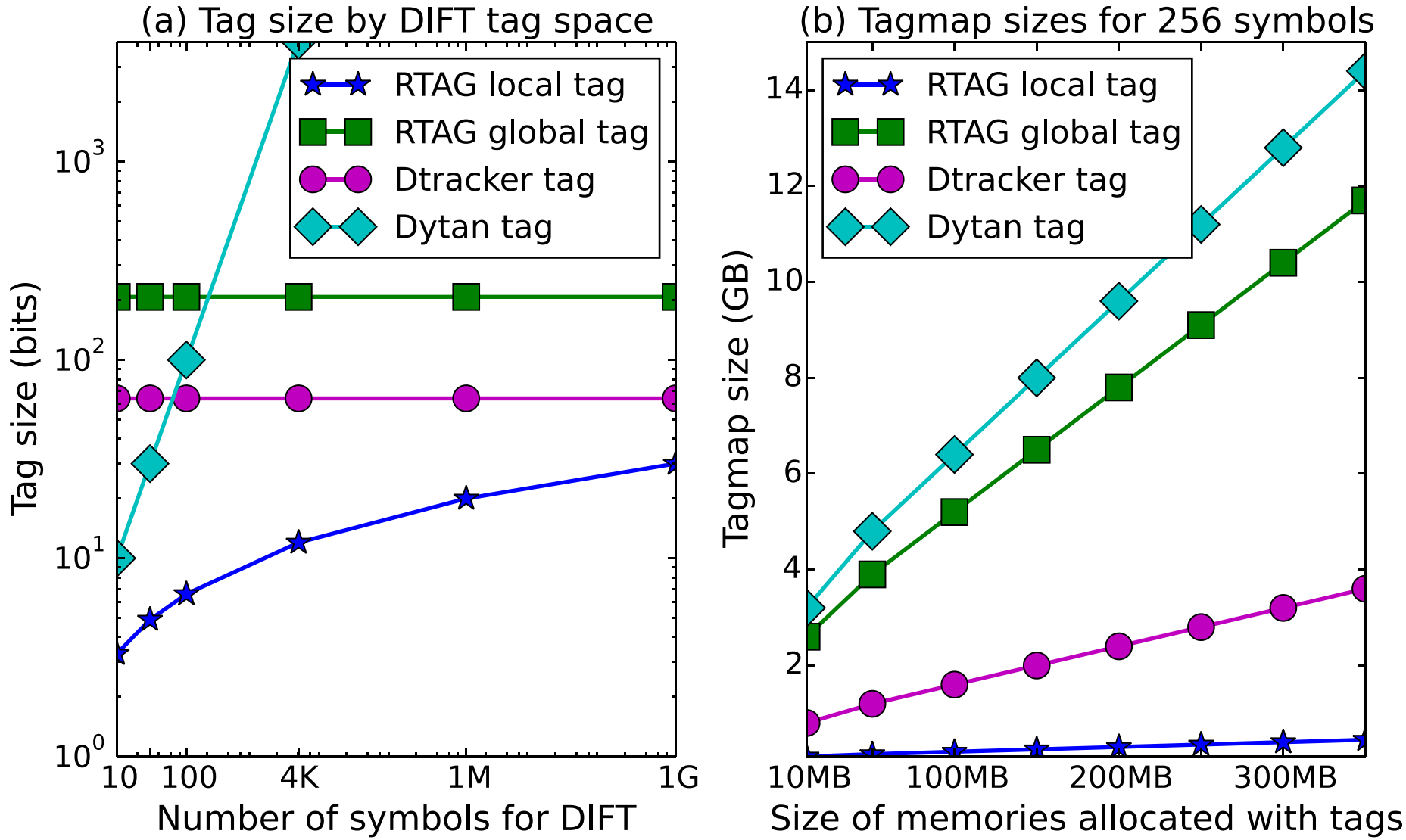
Tag allocation by analyzing syscall trace



Local tag: 0x0–0x5

~~208bits~~ → 3 bits

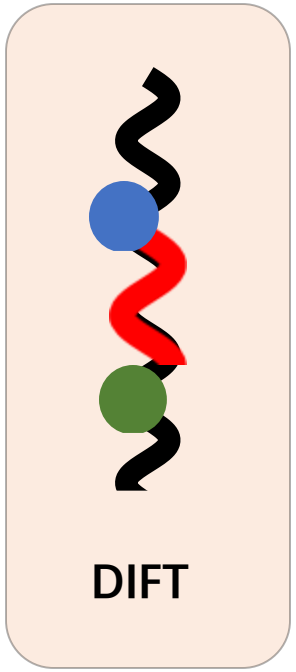
Comparison of tag sizes with previous systems



Tag switch at IO syscall entry and exit during DIFT

Global	
offset	key
0	U0 (208bit)
1	U1
2	U2

Local	
offset	key
0	0x0
1	0x1
2	0x2



Local	
offset	key
0	0x1
1	0x1
2	0x0

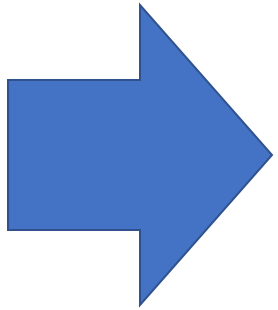
Global		
offset	key	value
0	U9	U1
1	U10	U1
2	U11	U0

Tag association

- Need to link the tag propagation between two hosts via socket communication
- Support both TCP and UDP packets with tag association but in different ways
  - TCP: counter-based
  - UDP: tag-embedding-based

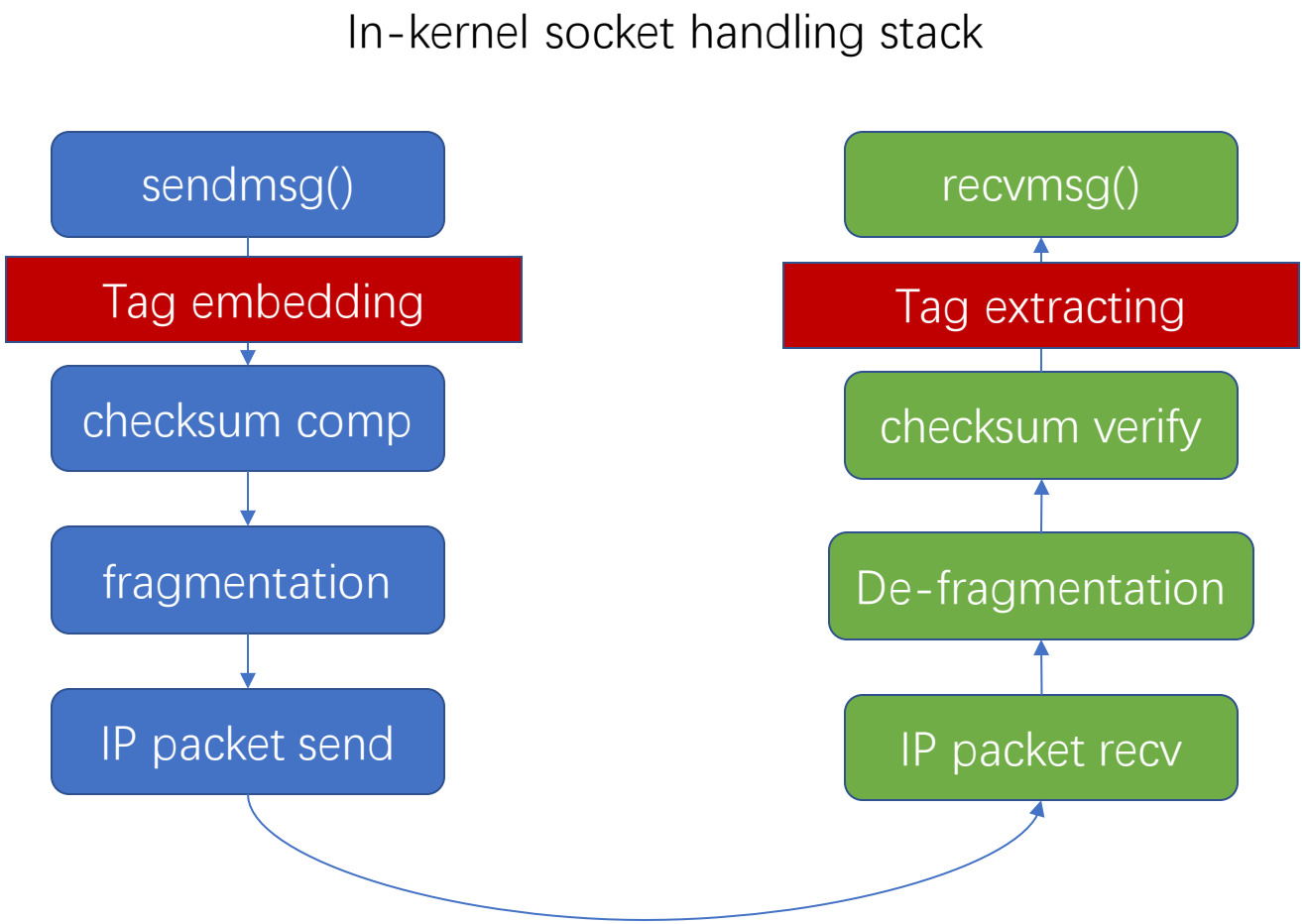
TCP (order-preserving transmission)

Syscall	Data offset	Offset in tag
send	0	0x0
	1	0x1
	2	0x2
	3	0x3
send	0	0x4
	1	0x5
send	0	0x6
	1	0x7
	2	0x8
	3	0x9
	4	0xA
	5	0xB

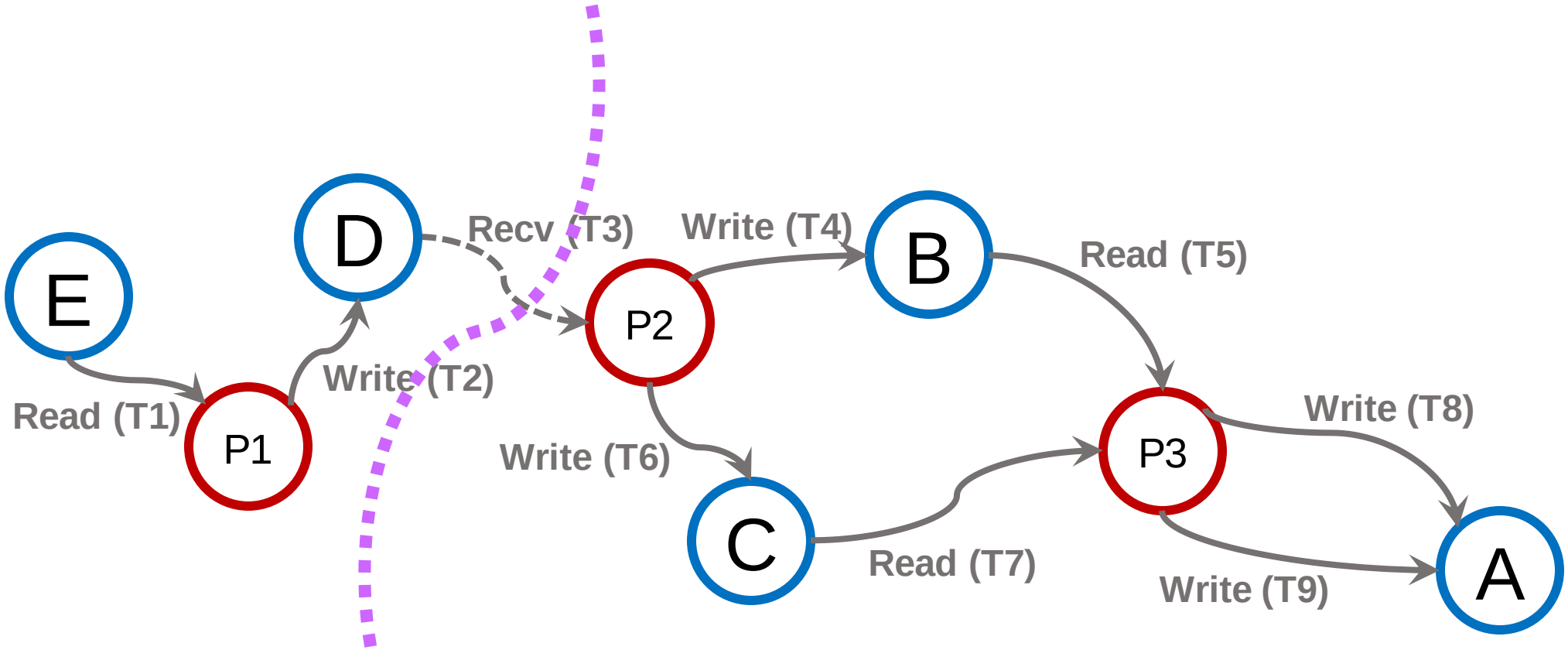


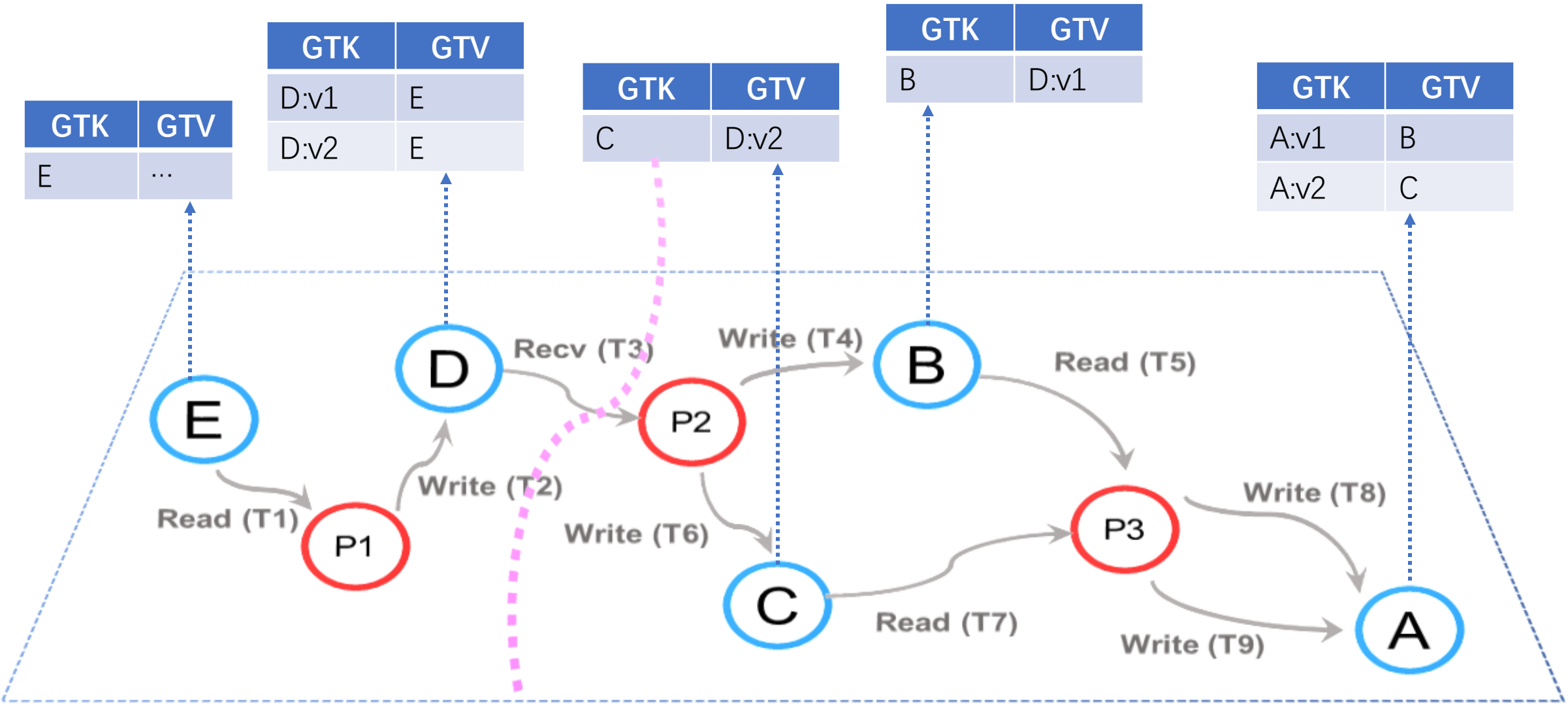
Syscall	Data offset	Offset in tag
read	0	0x0
	1	0x1
	2	0x2
read	0	0x3
	1	0x4
	2	0x5
read	0	0x6
	1	0x7
	2	0x8
read	0	0x9
	1	0xA
	2	0xB

UDP (datagram transmission)

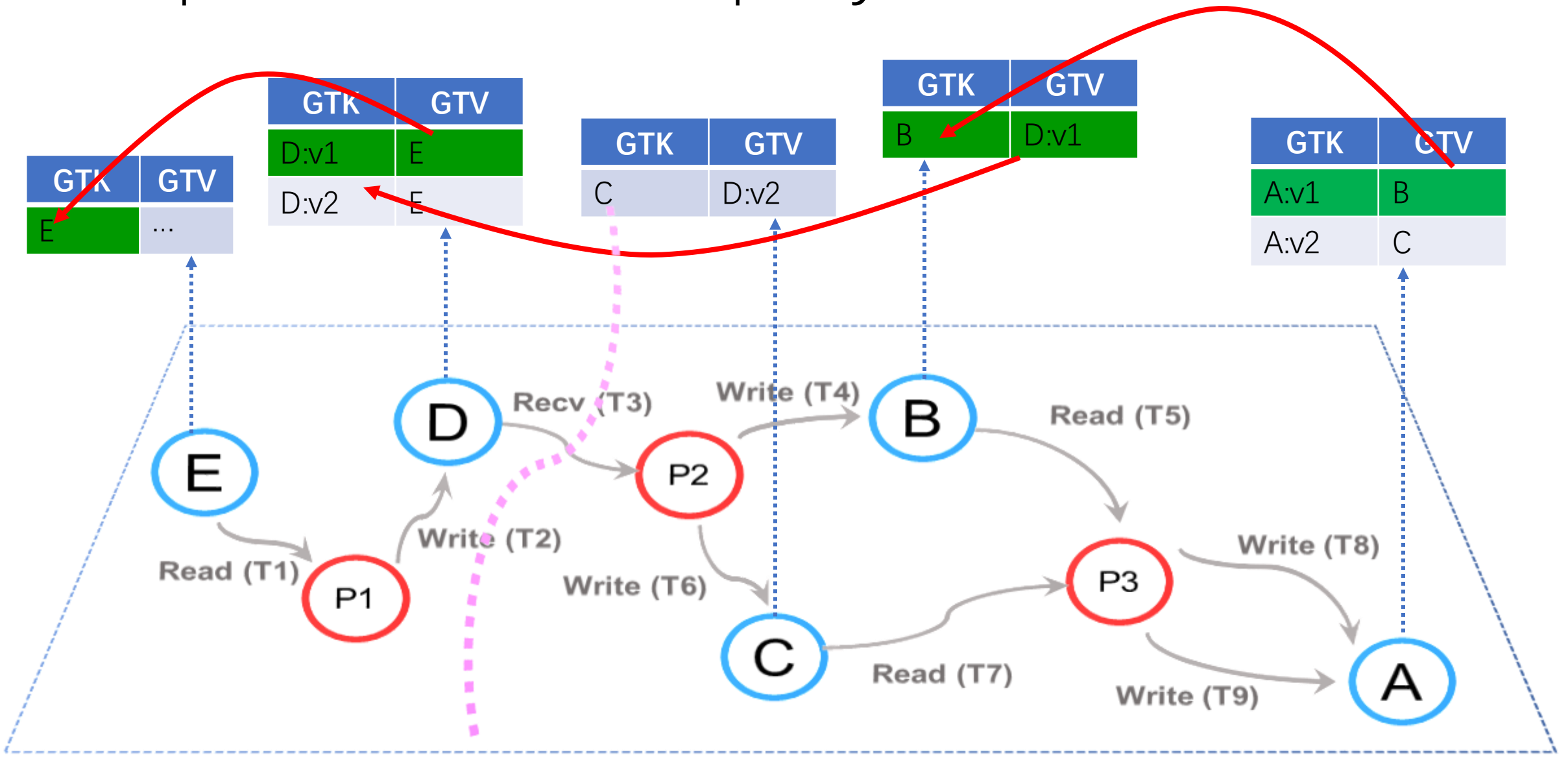


Tag overlay on top of provenance graph




GTK
GTV
E
…
GTK
GTV
D:v1
E
D:v2
E
GTK
GTV
C
D:v2
GTK
GTV
B
D:v1
GTK
GTV
A:v1
B
A:v2
C
E
D
P2
B
P1
P3
C
A
Recv (T3)
Write (T4)
Read (T5)
Read (T1)
Write (T2)
Write (T6)
Write (T8)
Read (T7)
Write (T9)

Example of backward query from A:v1



Platform and dependencies

- Run on Ubuntu 12.04 LTS 32-bit and 64-bit
- Use libdft as DIFT propagation engine (32-bit and 64-bit)
- Use Neo4j for graph-based reachability analysis
- Use PostgreSql for tag storage

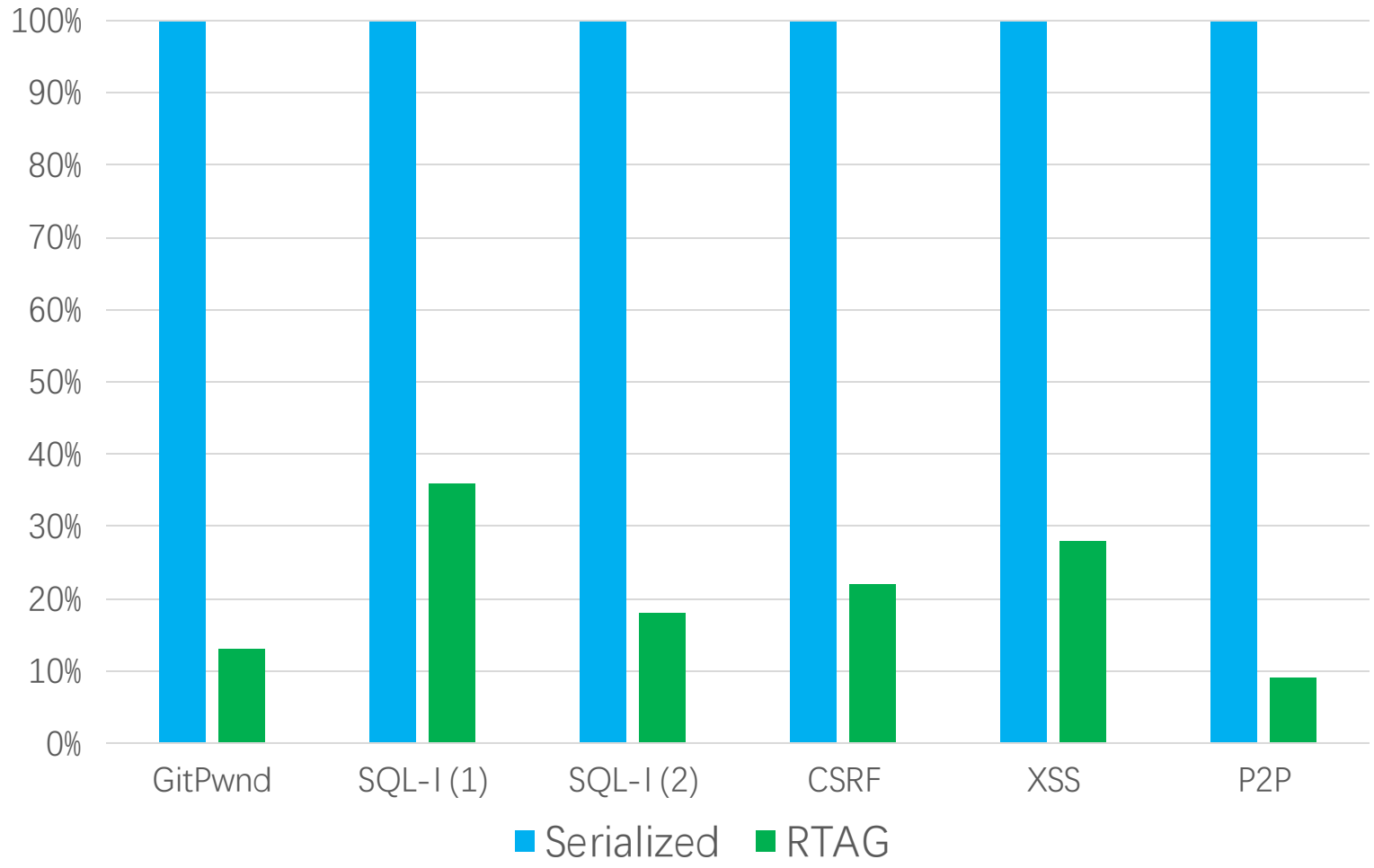
Evaluation

- Effectiveness
- Analysis overhead
- Runtime overhead

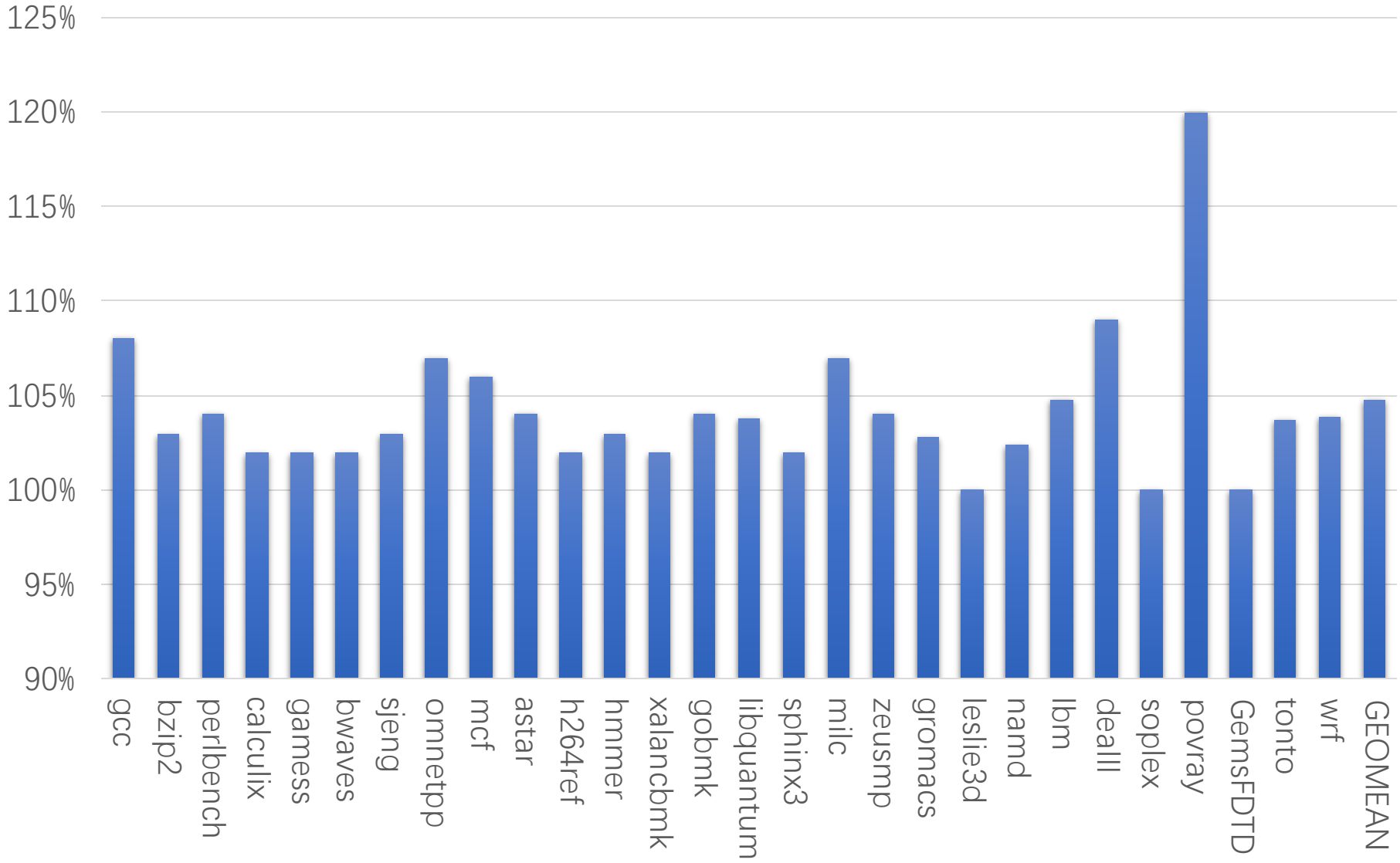
Effectiveness

Attack	Ex query	Accurate?
GitPwnd (git, gitolite)	Forward: /etc/passwd	√
SQL-I (1) (Firefox, Apache)	Backward: payroll record	√
SQL-I (2) (same as above)	Backward: dump file	√
CSRF (same as above)	Forward: exploit html	√
XSS (same as above)	Point-to-point: html –attack_host	√
P2P (6 hosts, gnutella)	Forward: mp4@1st node	√

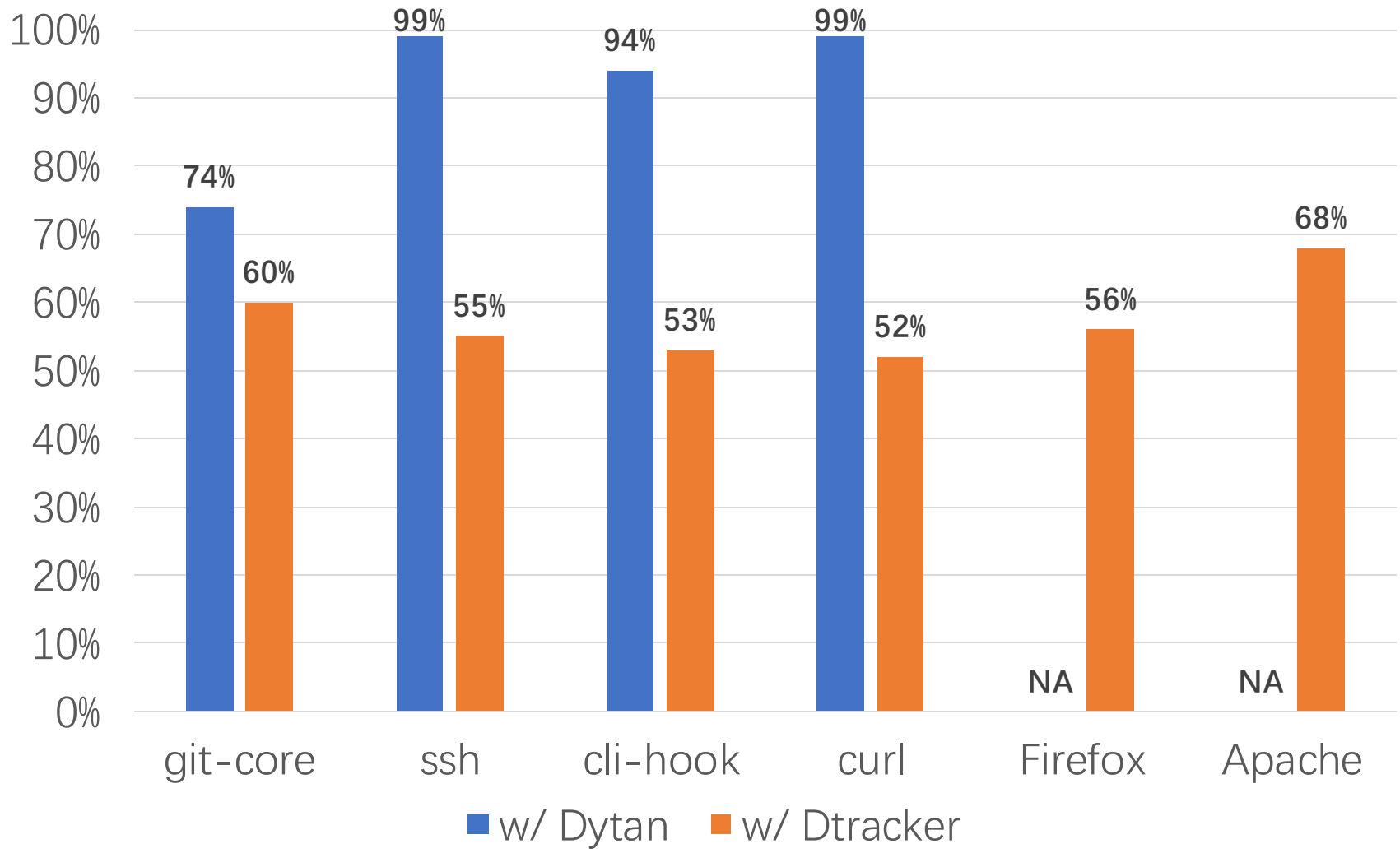
Analysis time reduction ~90%



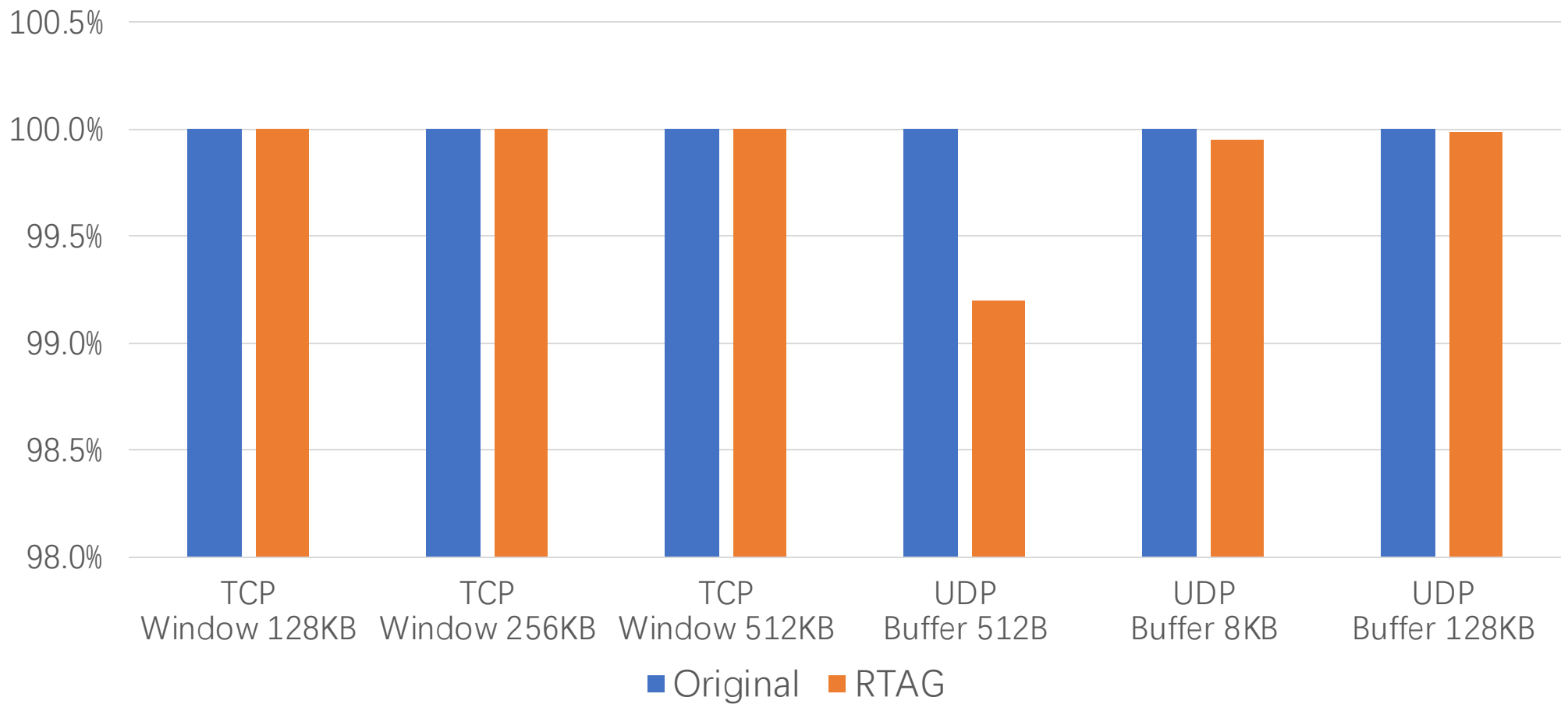
Runtime overhead: 4.84% SPEC CPU2006



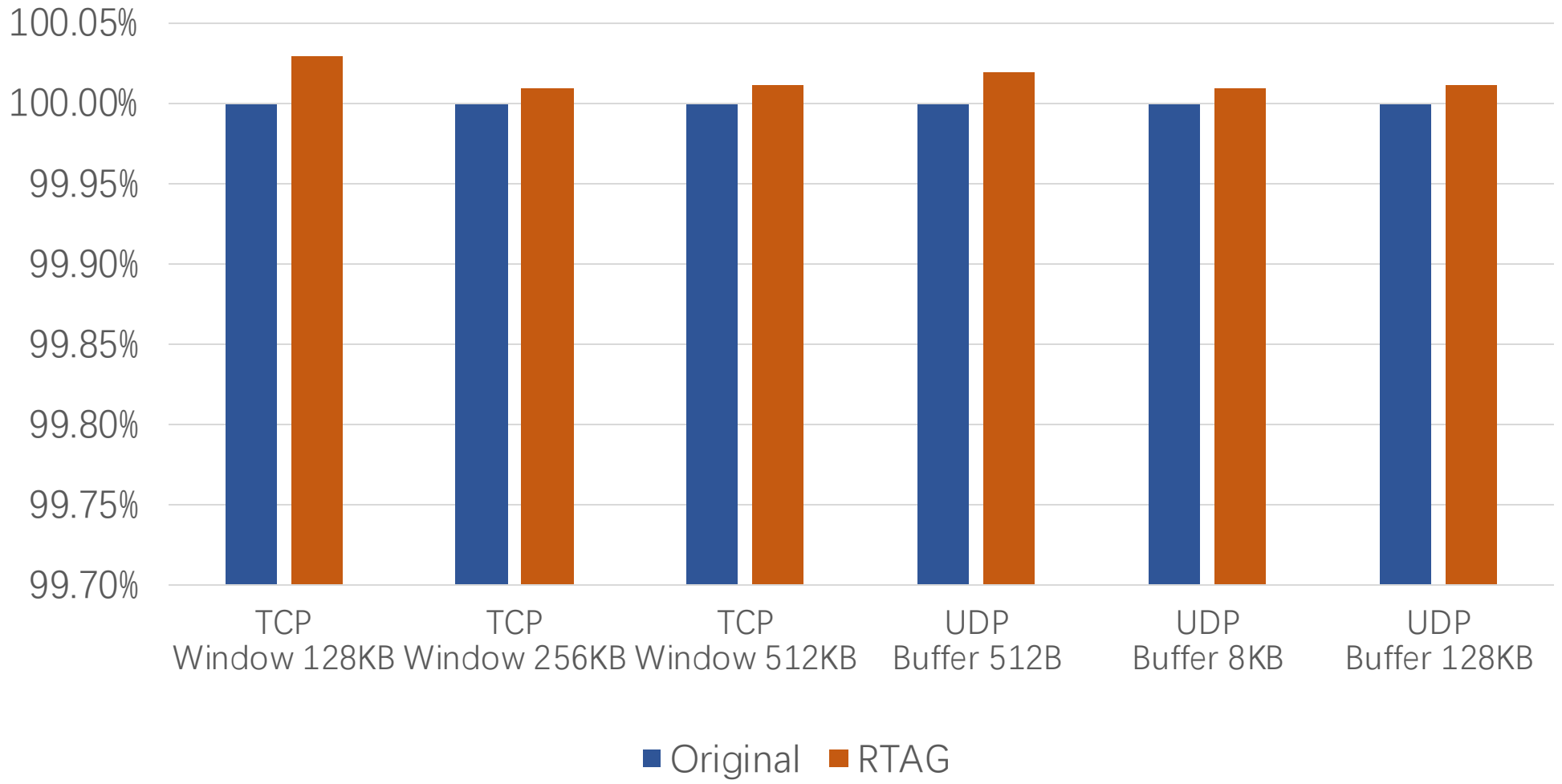
Memory cost reduction: 50%–99%



Network Impact: bandwidth <0.5% reduction



Network impact: Round-trip-time <0.05% increase



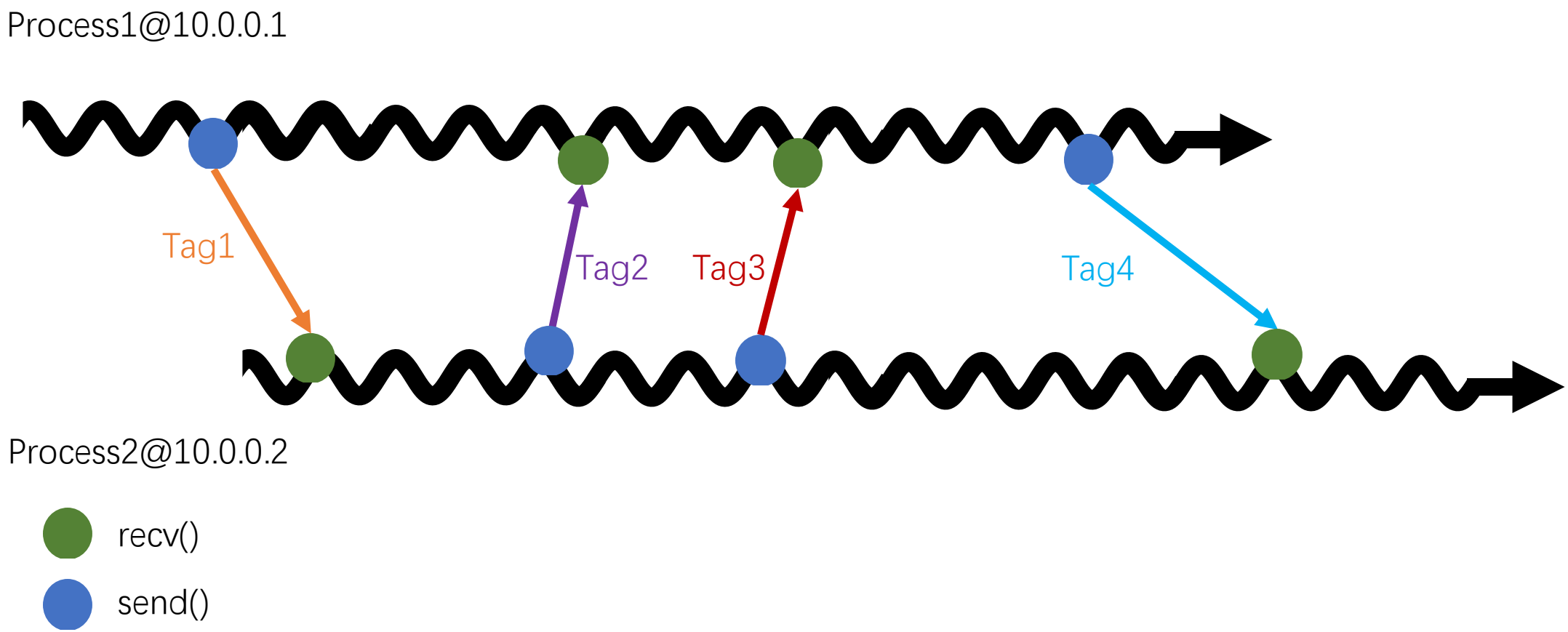
Conclusion

- RTAG enables the cross-host refinable attack investigation
  - Decouple the tag dependency from the replayed DIFT
  - Optimally allocate tags for each DIFT based on reachability analysis

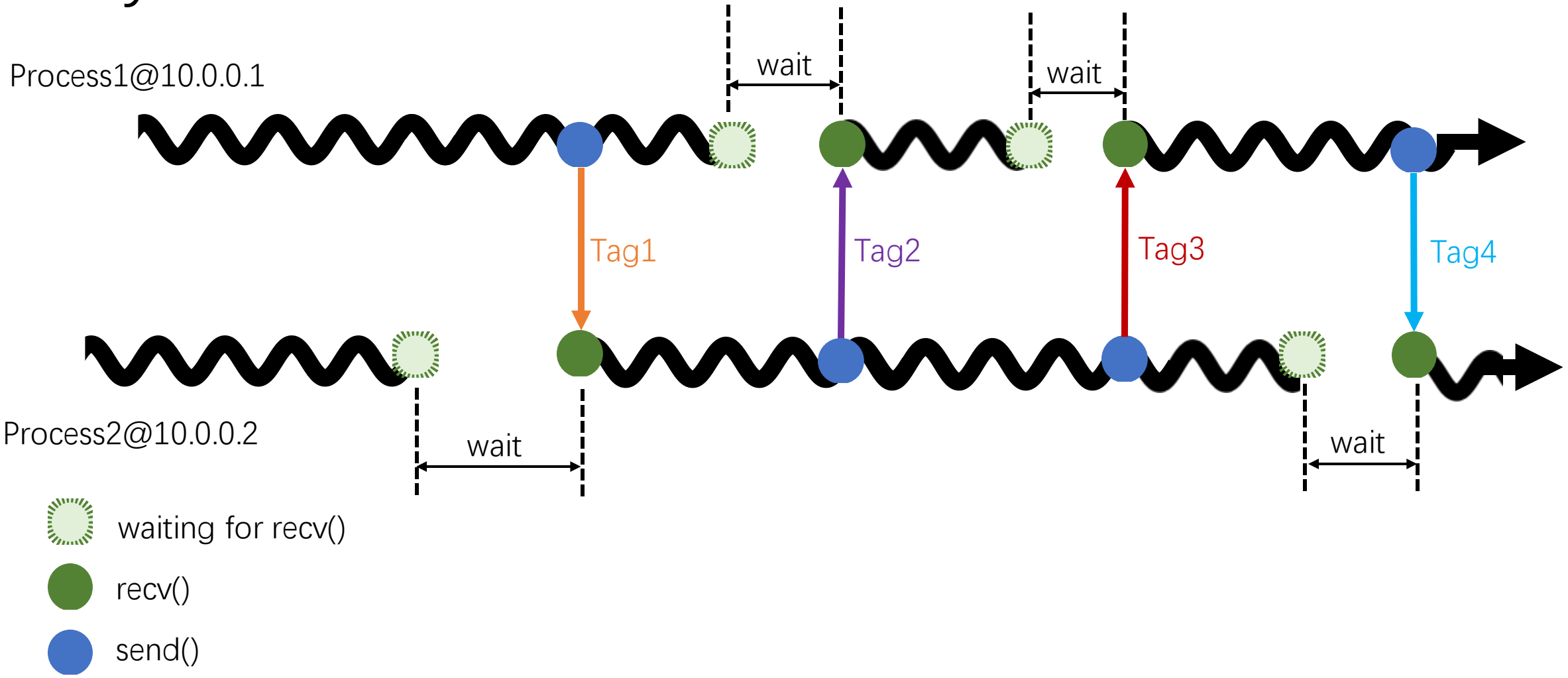
- RTAG achieves good performance
  - Runtime: run with negligible overhead (<5%)
  - Analysis: reduce analysis time cost by 60%–90%, memory cost by up to 90%

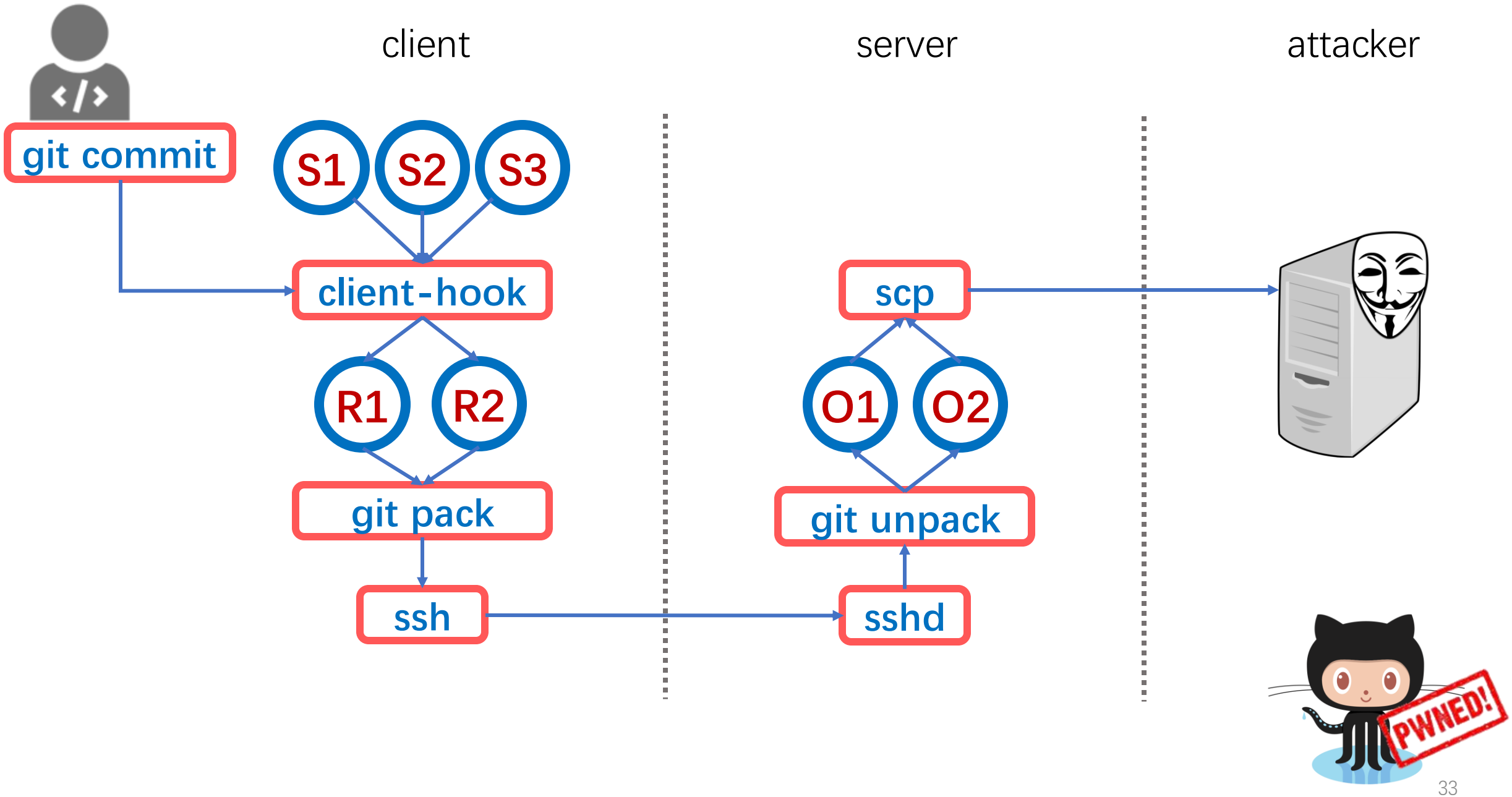
Back up slides

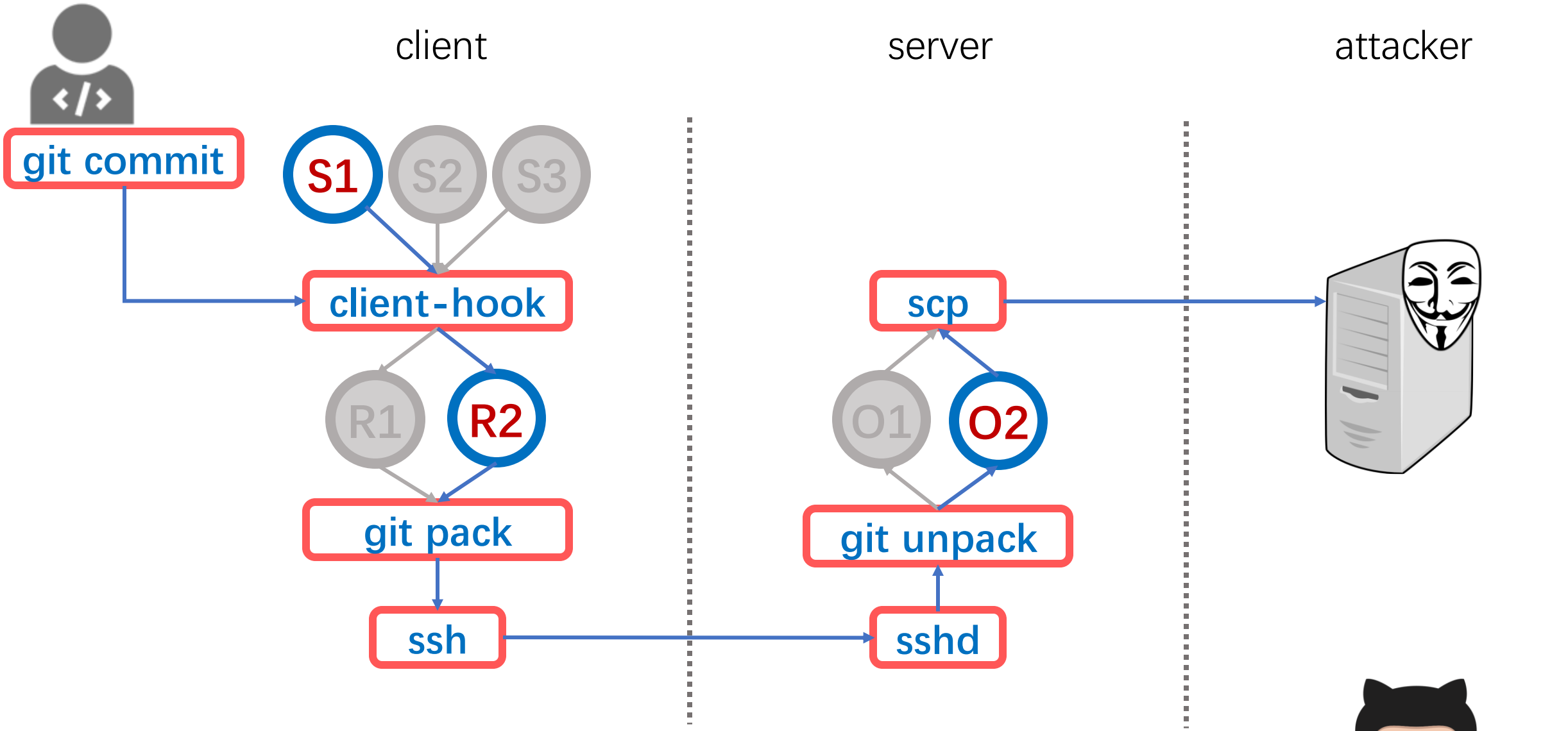
Example of serialization due to tag dependencies



Because replayed processes are not synchronized




client
server
attacker
git commit
S1
S2
S3
client-hook
R1
R2
git pack
ssh
scp
O1
O2
git unpack
sshd
PWNED!



Answer queries:

- What data were leaked to the attacker?
- Was S2 leaked?
- How was S1 leaked step in step?