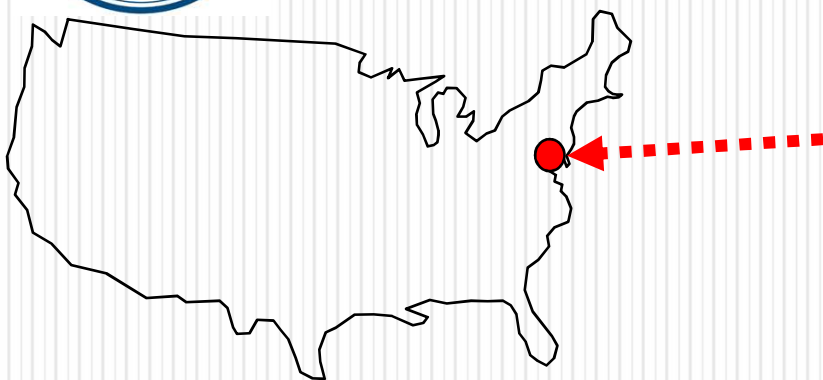


Misbehaviors in TCP SACK Generation



Nasif Ekiz, Abuthahir H. Rahman,
Paul D. Amer



Protocol Engineering Laboratory
Computer and Information Sciences,
University of Delaware

supported by  **CISCO**

OUTLINE

1. Introduction
2. Experimental design
3. 7 SACK misbehaviors
4. Results
5. Summary

OUTLINE

1. Introduction
2. Experimental design
3. 7 SACK misbehaviors
4. Results
5. Summary

TCP acknowledgements

- cumulative ACK n (for ordered data)
bytes $[... \text{ to } n-1]$ [RFC 793]
- selective ACK (SACK) $m-n$ (for out-of-order data)
bytes $[m \text{ to } n-1]$ [RFC 2018]
 - prevents unnecessary retransmissions during loss recovery
 - improves throughput when multiple losses in same window
(SACK-based loss recovery [RFC 3517])

Deployment of SACK

- **2001:** 41% of web servers do SACK
- **2005:** 68% of web servers do SACK
- **2010:** All recent versions of OSes do SACK

➤ FreeBSD



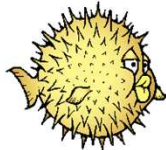
➤ Linux



➤ Mac OS X



➤ OpenBSD



➤ Open Solaris



➤ Solaris



opensolaris™

➤ Windows



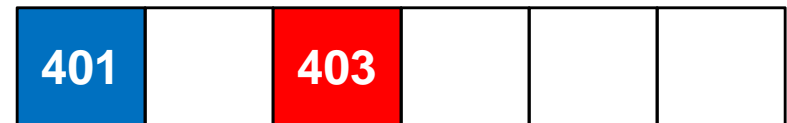
Data renegeing

receive buffer

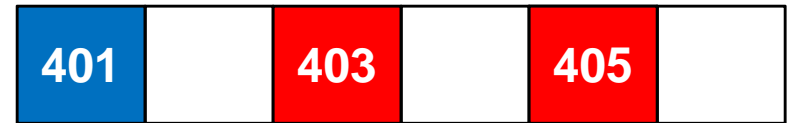
ACK 402



ACK 402 SACK 403-404



ACK 402 SACK 405-406 403-404



ACK 402



ordered data (ACKed)



out-of-order data (SACKed)



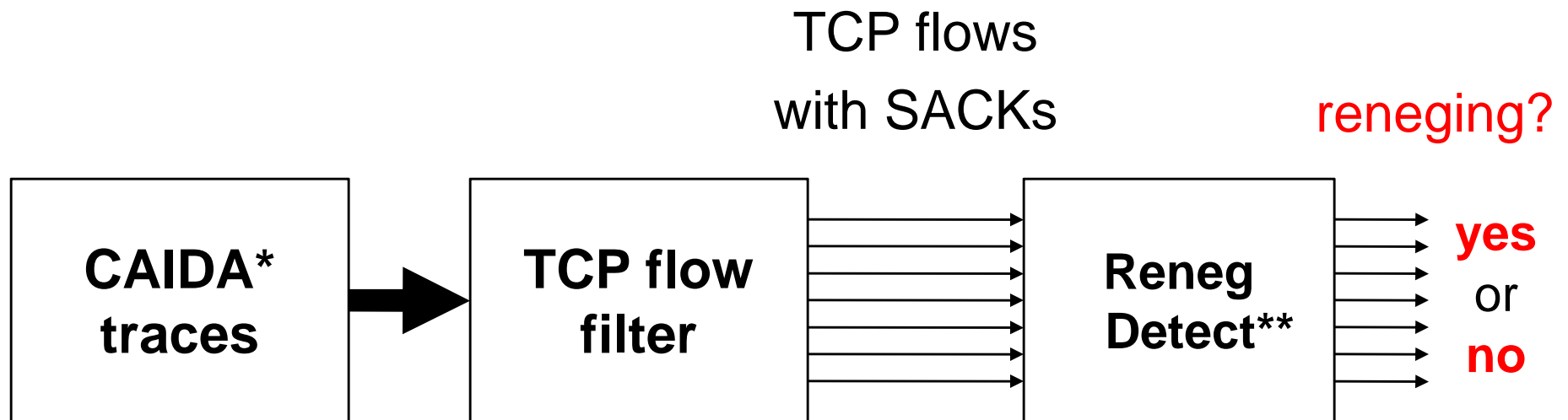
available space



Data renegeing

- TCP is designed to tolerate renegeing
 - [RFC 2018]: “The SACK option is *advisory*, in that, while it notifies the data sender that the data receiver has received the indicated segments, the data receiver is permitted to later *discard* data which have been reported in a SACK option.”
 - TCP data sender retains copies of all SACKed data until ACKed

RenegDetect



* Cooperative Association for Internet Data Analysis

** Nasif Ekiz, Paul D. Amer, "***A model for detecting transport layer data reneging***", PFLDNeT 2010, Lancaster, PA, 11/10

RenegDetect

- RenegDetect was tested with **real** TCP flows
 - at first, renegeing seemed to occur **frequently**
 - on closer inspection, we found many **unexpected** behaviors !
 - sometimes SACKs that should have been sent were not
 - sometimes the wrong SACKs were sent
- these SACK misbehaviors wrongly gave the impression that **data renegeing** was occurring

Research goal

- verify TCP SACK for a wide-range of OSes
- **not** measure degraded performance
 - misbehaviors can reduce the effectiveness of SACKs

OUTLINE

1. Introduction
2. Experimental design
3. 7 SACK misbehaviors
4. Results
5. Summary

Experimental design

TCP Behavior Inference Tool

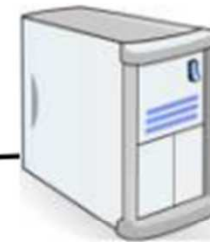
TCP Implementation Under Test

TBIT



LAN

TCP IUT



FreeBSD 7.1

TBIT 1.0

Ubuntu 9.10

VirtualBox

FreeBSD Linux Windows

OpenBSD Solaris

Apache HTTP Server

Tcpdump/Wireshark

OUTLINE

1. Introduction
2. Experimental design
3. **7 SACK misbehaviors**
4. Results
5. Summary

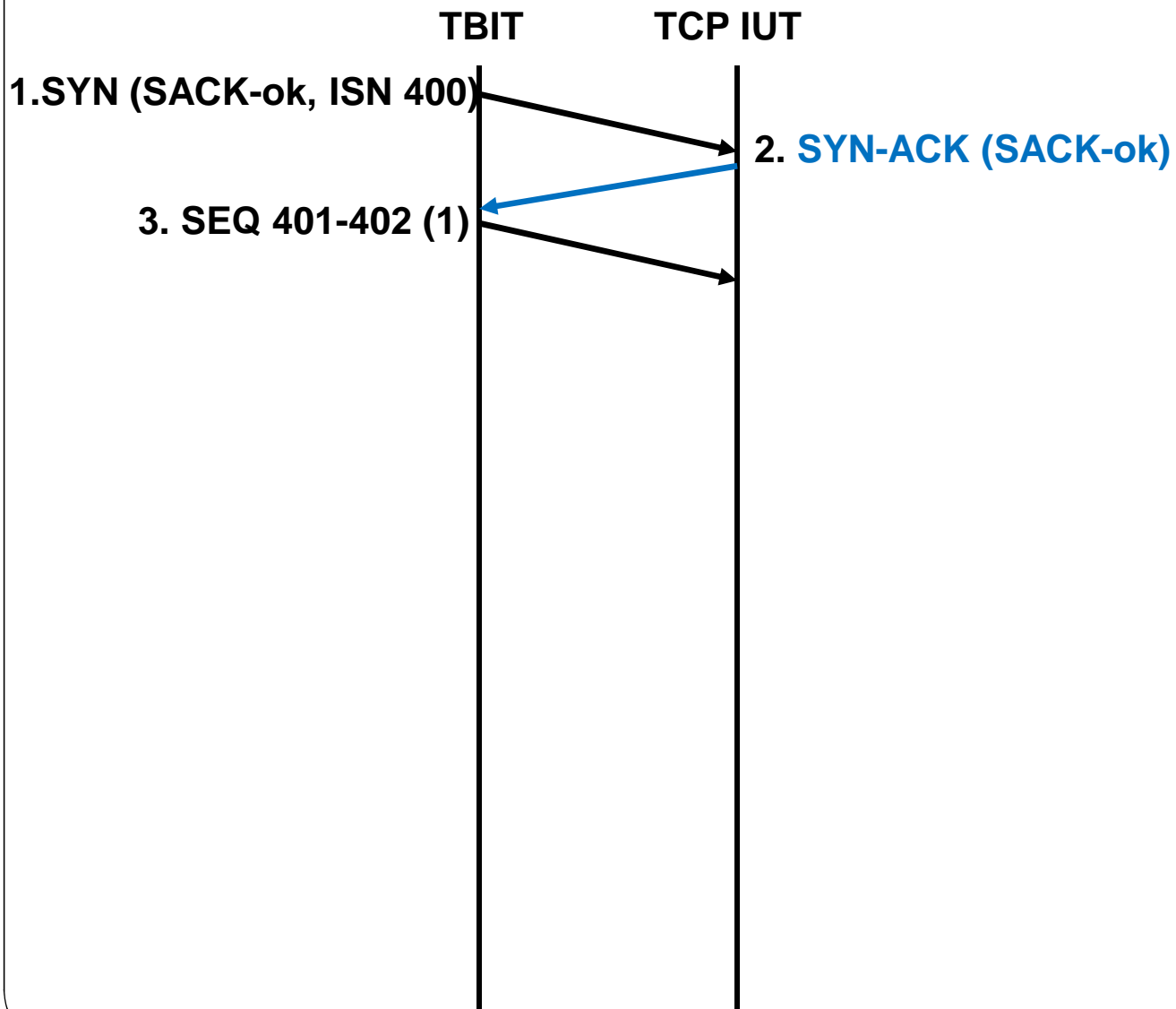
MISBEHAVIOR A

Misbehavior A:

fewer than maximum number of reported SACKs

- RFC 2018 section 3
 - “the data receiver *SHOULD* include *as many distinct SACK blocks possible in the SACK option,*” and
 - “the 40 bytes available for TCP options can specify a *maximum of four SACK blocks*”
- for some TCP flows, only 2 or 3 SACK blocks are reported even when 4 are possible

Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs

receiving
application

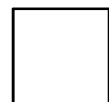
receive buffer



ordered data (ACKed)

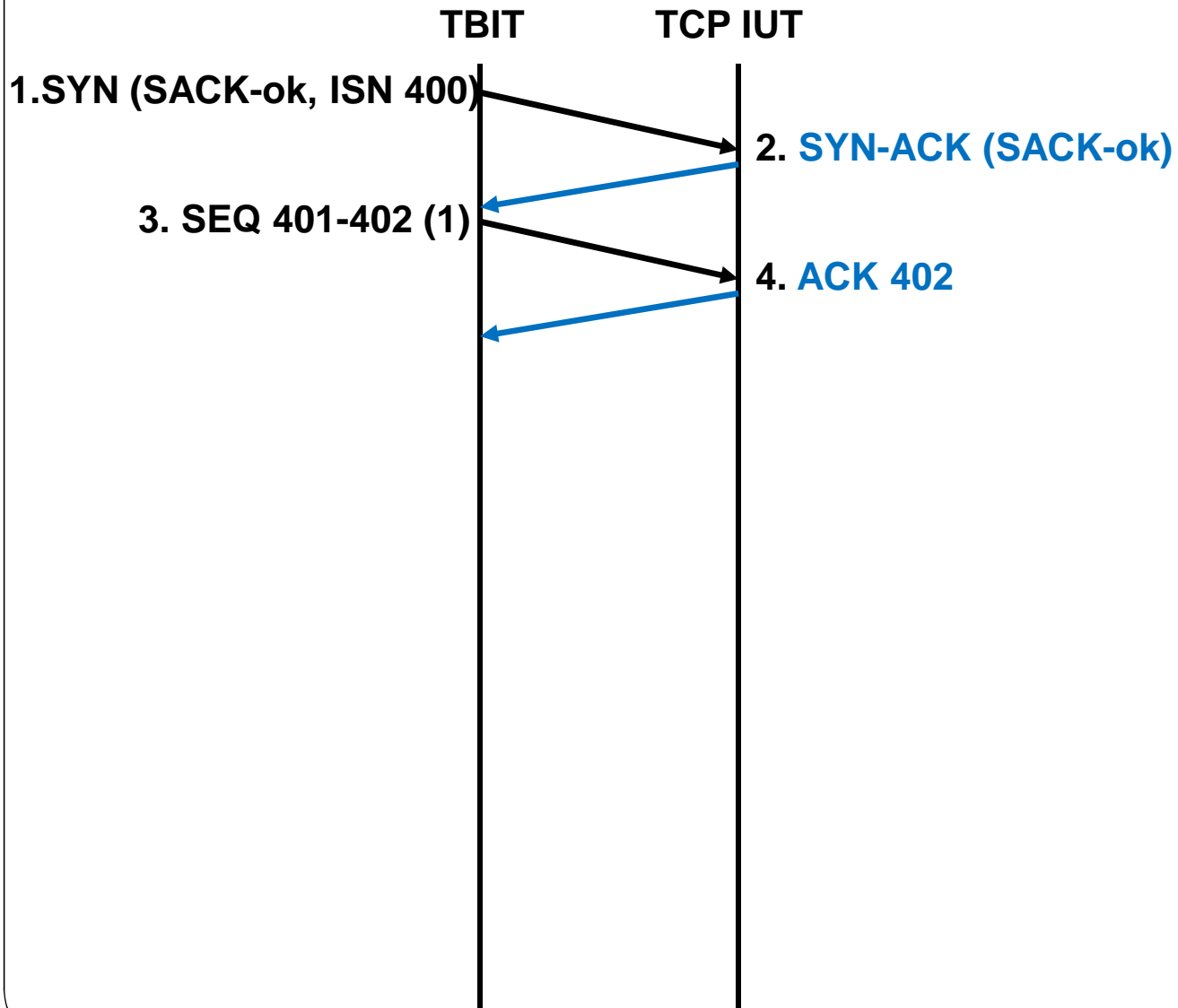


out-of-order data (SACKed)

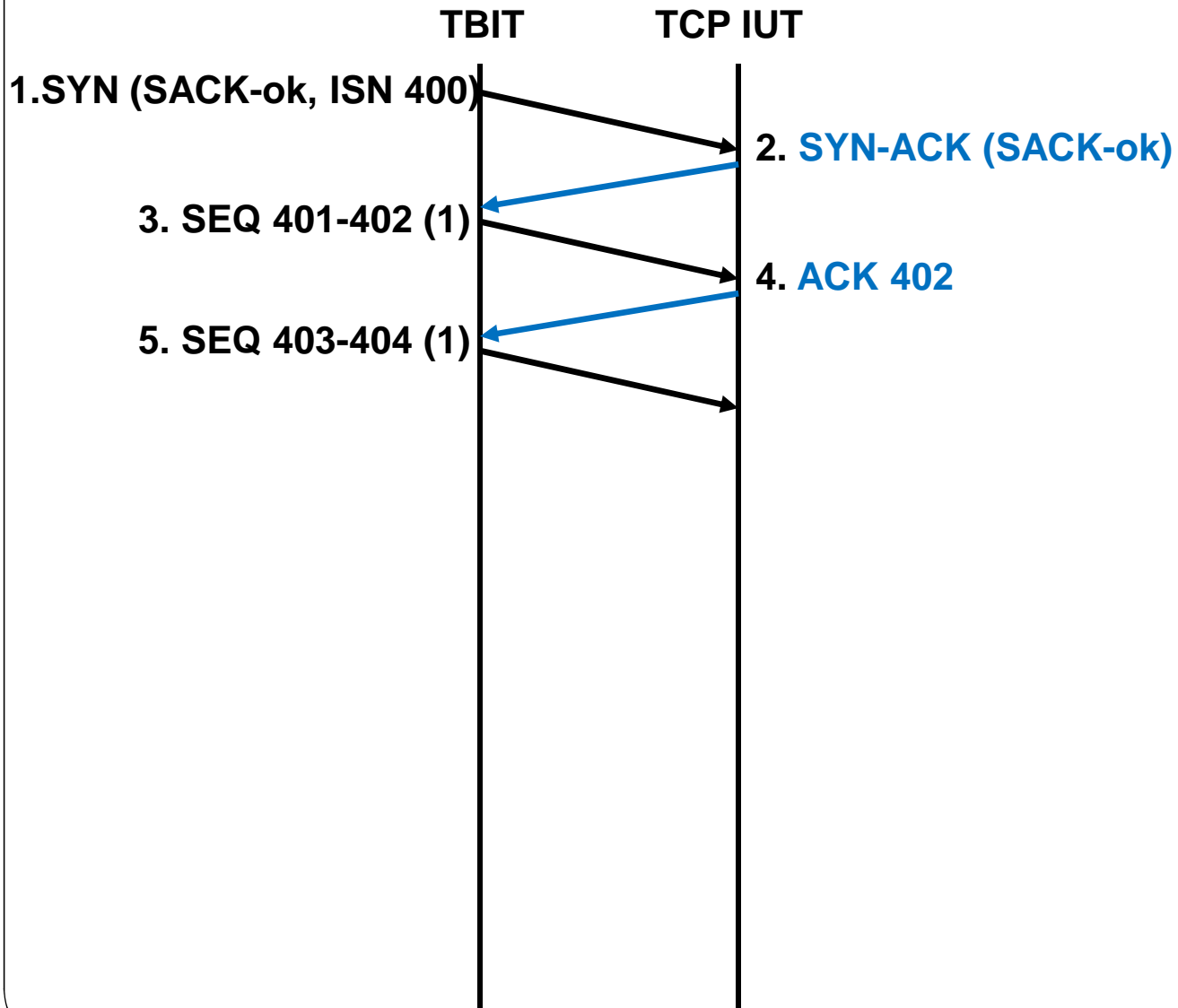


available space

Misbehavior A: fewer than maximum number of reported SACKs



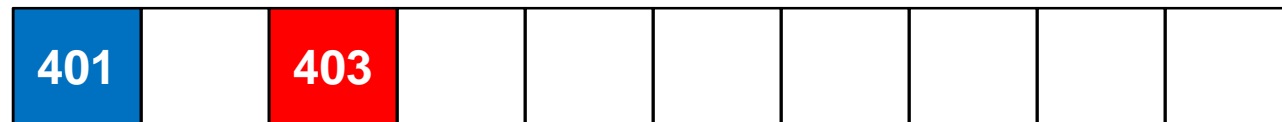
Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs

receiving
application

receive buffer



ordered data (ACKed)

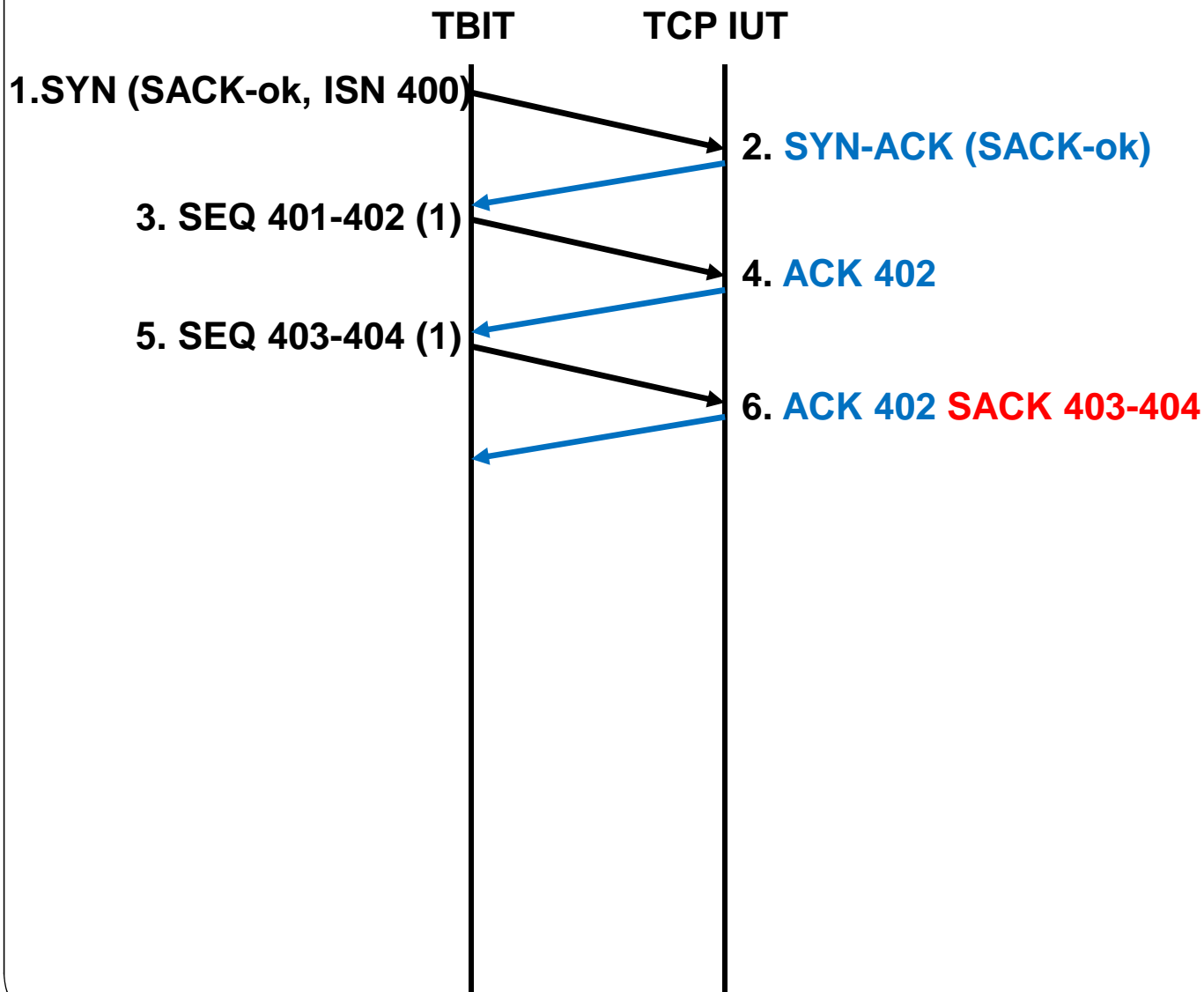


out-of-order data (SACKed)

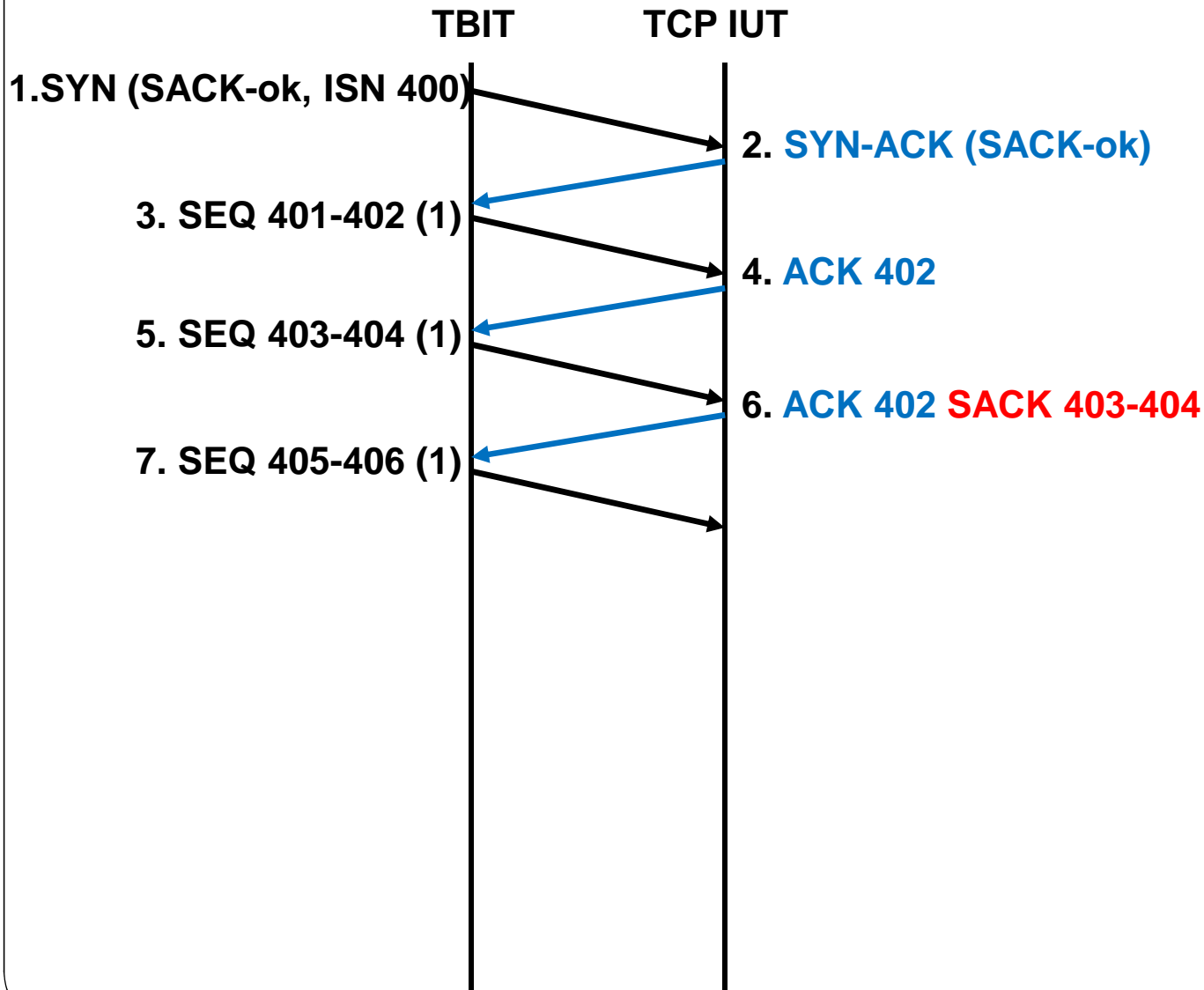


available space

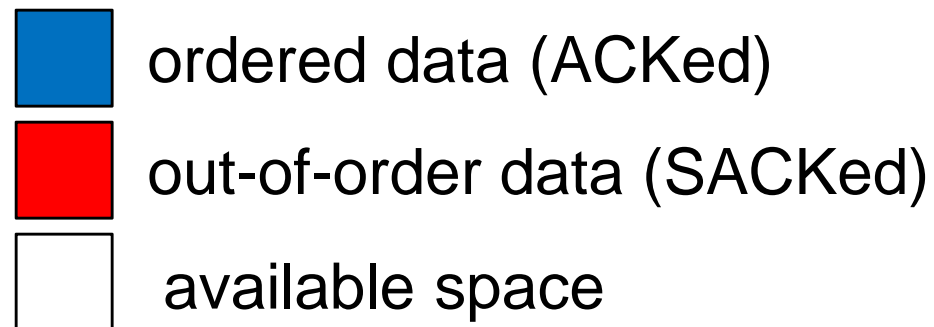
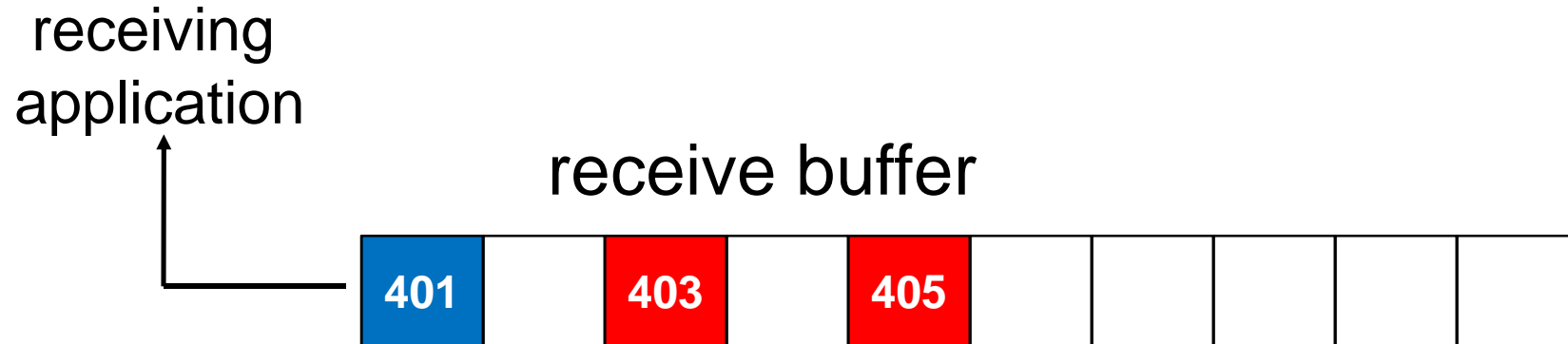
Misbehavior A: fewer than maximum number of reported SACKs



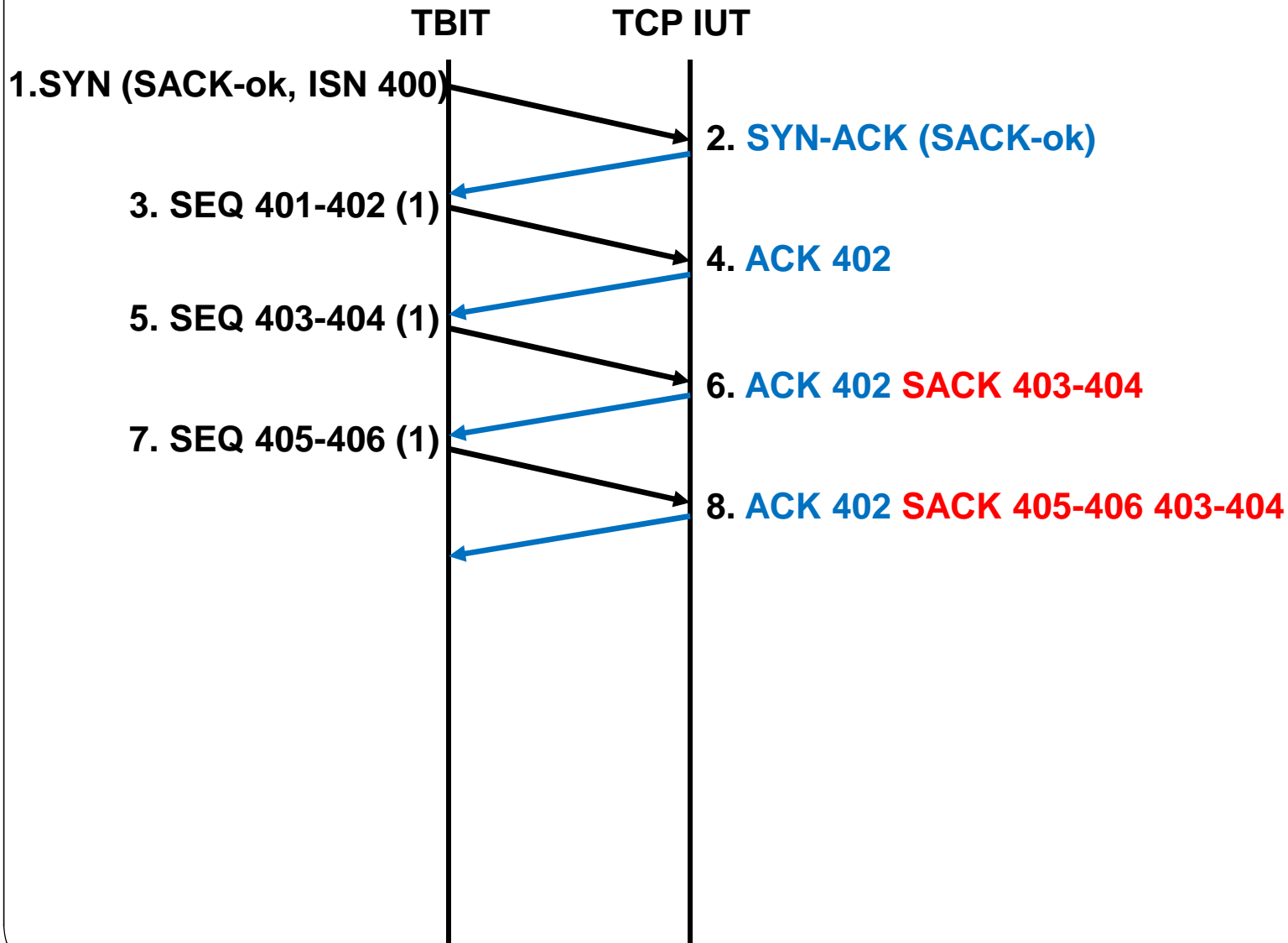
Misbehavior A: fewer than maximum number of reported SACKs



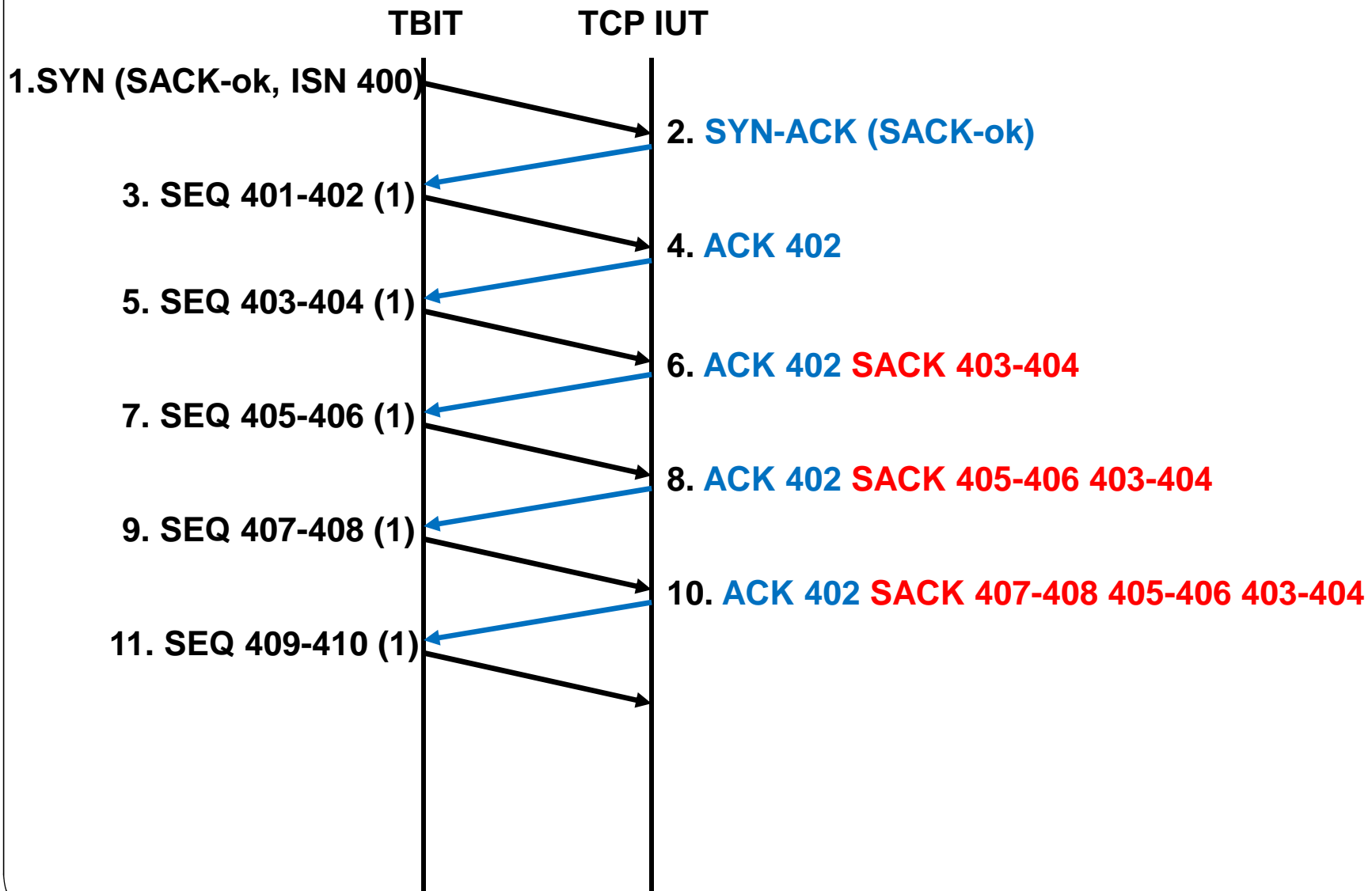
Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs



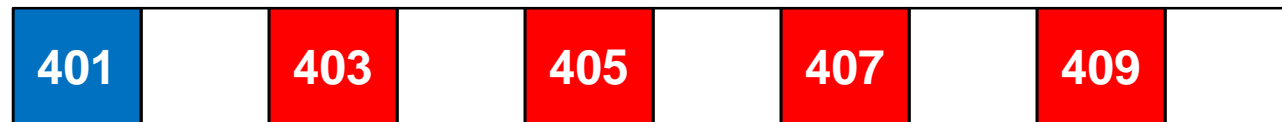
Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs

receiving
application

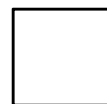
receive buffer



ordered data (ACKed)

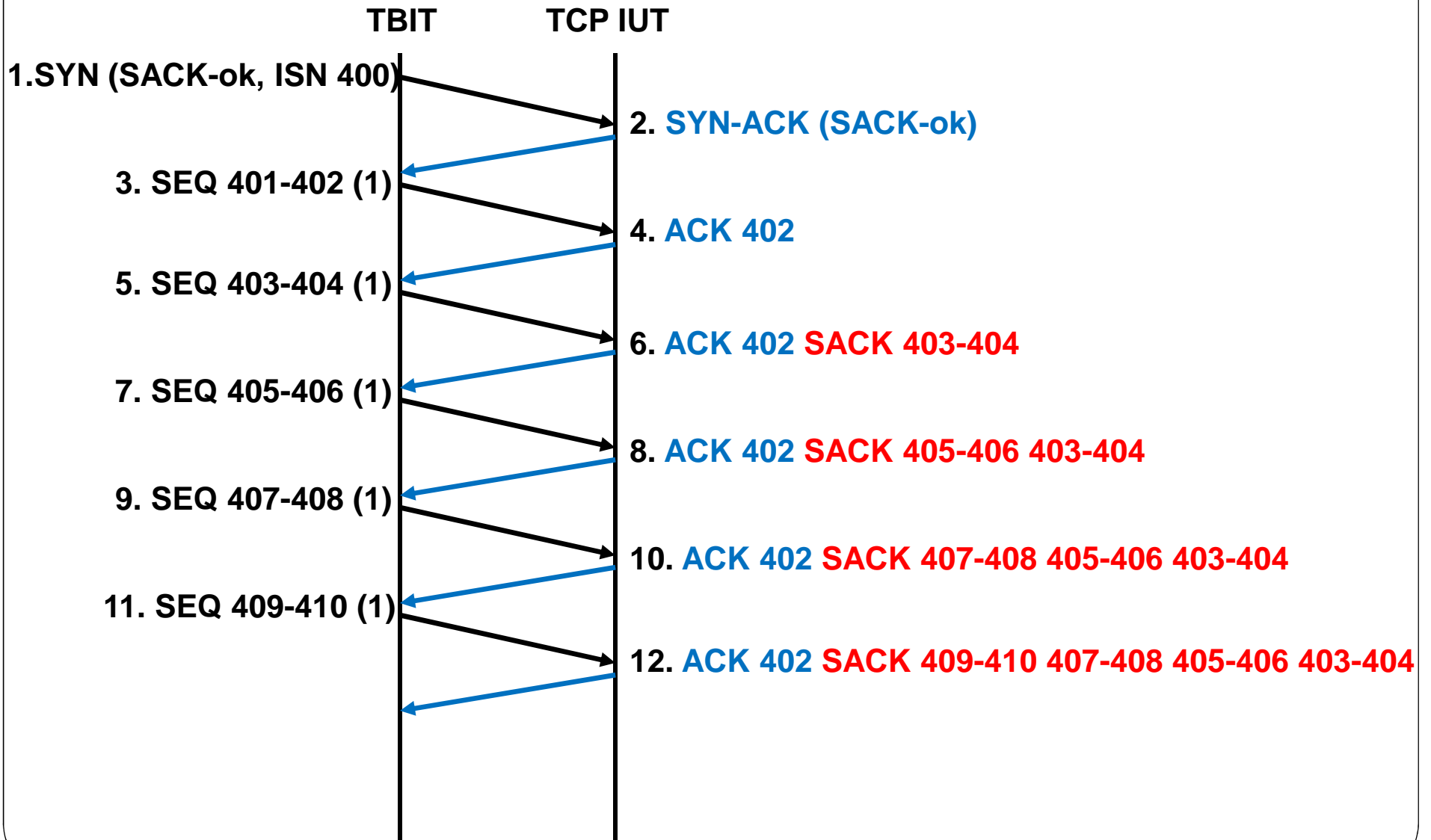


out-of-order data (SACKed)

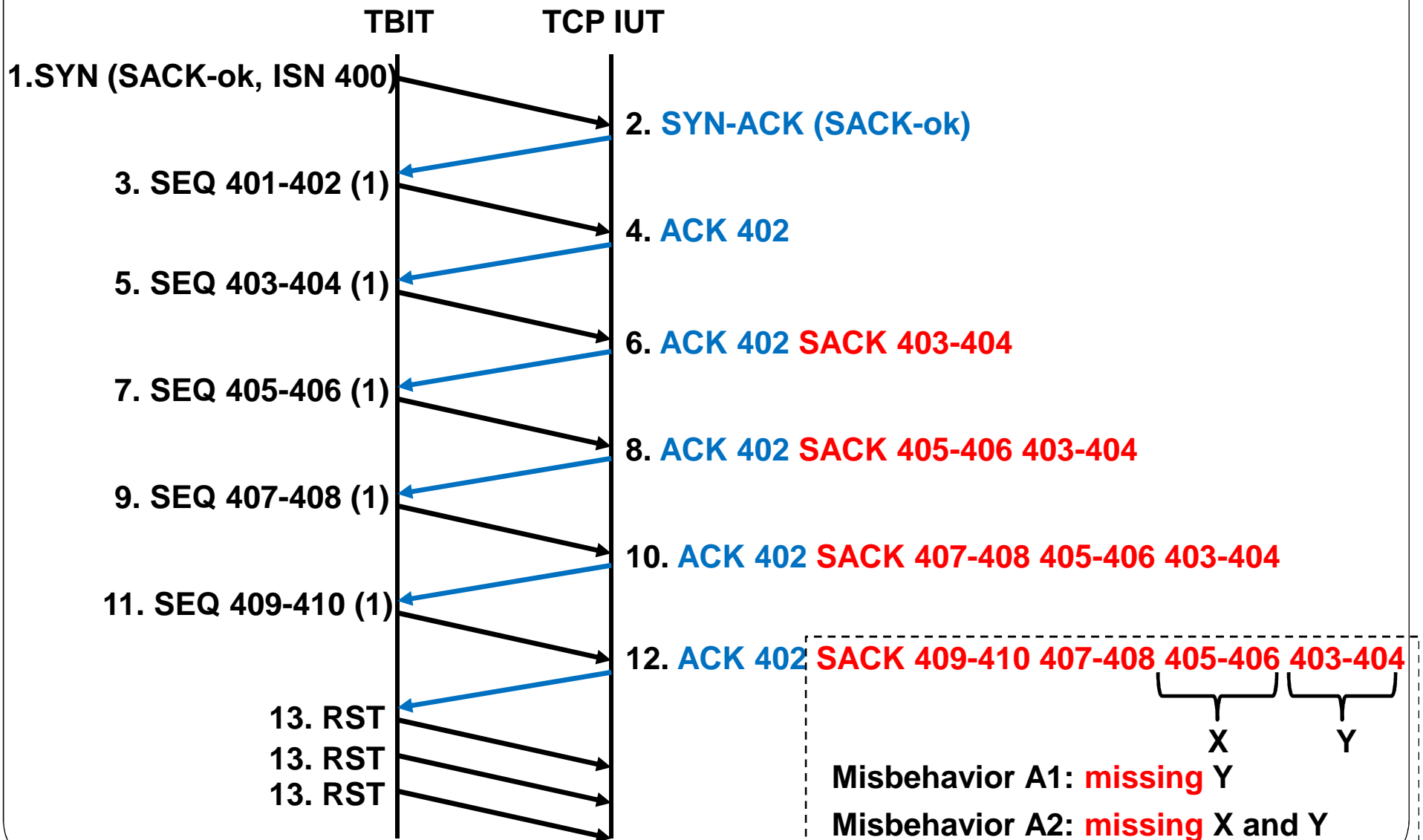


available space

Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs



Misbehavior A: fewer than maximum number of reported SACKs

26	01:15:16:179012	128.4.30.30	128.4.30.12	TCP	[TCP Dup ACK 18#4] http > irisa [ACK] Seq=2742779
27	01:15:16:194864	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0
28	01:15:16:197054	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0
29	01:15:16:198673	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0

```
*****
▶ Frame 26 (74 bytes on wire, 74 bytes captured)
▶ Ethernet II, Src: Xensourc_0d:23:37 (00:16:3e:0d:23:37), Dst: Dell_0d:2b:14 (00:13:72:0d:2b:14)
▶ Internet Protocol, Src: 128.4.30.30 (128.4.30.30), Dst: 128.4.30.12 (128.4.30.12)
▼ Transmission Control Protocol, Src Port: http (80), Dst Port: irisa (11000), Seq: 2742779501, Ack: 2881284402, Len: 0
  Source port: http (80)
  Destination port: irisa (11000)
  Sequence number: 2742779501
  Acknowledgement number: 2881284402
  Header length: 40 bytes
  ▶ Flags: 0x10 (ACK)
  Window size: 65534
  ▶ Checksum: 0xe782 [correct]
  ▼ Options: (20 bytes)
    NOP
    NOP
    ▶ SACK 2881284409-2881284410 2881284407-2881284408
  ▶ [SEQ/ACK analysis]
```

0000	00 13 72 0d 2b 14 00 16 3e 0d 23 37 08 00 45 00	...r+. >.#7..E.
0010	00 3c 00 2c 40 00 80 06 be 5d 80 04 1e 1e 80 04	<.,@...].....
0020	1e 0c 00 50 2a f8 a3 7b 7e 6d ab bc e9 32 a0 10	...P*..{ ~m...2..
0030	ff fe e7 82 00 00 01 01 05 12 ab bc e9 39 ab bc9..
0040	e9 3a ab bc e9 37 ab bc e9 387..8

Misbehavior A: fewer than maximum number of reported SACKs

26	01:15:16.179012	128.4.30.30	128.4.30.12	TCP	[TCP Dup ACK 18#4] http > irisa [ACK] Seq=2742779
27	01:15:16.194864	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0
28	01:15:16.197054	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0
29	01:15:16.198673	128.4.30.12	128.4.30.30	TCP	irisa > http [RST] Seq=2881284411 Win=7300 Len=0

▶ Frame 26 (74 bytes on wire, 74 bytes captured)

▶ Ethernet II, Src: Xensourc_0d:23:37 (00:16:3e:0d:23:37), Dst: Dell_0d:2b:14 (00:13:72:0d:2b:14)

▶ Internet Protocol, Src: 128.4.30.30 (128.4.30.30), Dst: 128.4.30.12 (128.4.30.12)

▼ Transmission Control Protocol, Src Port: http (80), Dst Port: irisa (11000), Seq: 2742779501, Ack: 2881284402, Len: 0

- Source port: http (80)
- Destination port: irisa (11000)
- Sequence number: 2742779501
- Acknowledgement number: 2881284402
- Header length: 40 bytes
- ▶ Flags: 0x10 (ACK)
- Window size: 65534
- ▶ Checksum: 0xe782 [correct]
- ▼ Options: (20 bytes) ← only 20 of possible 40 option bytes are used!
 - NOP
 - NOP
 - ▶ SACK: 2881284409-2881284410 2881284407-2881284408
- ▶ [SEQ/ACK analysis]

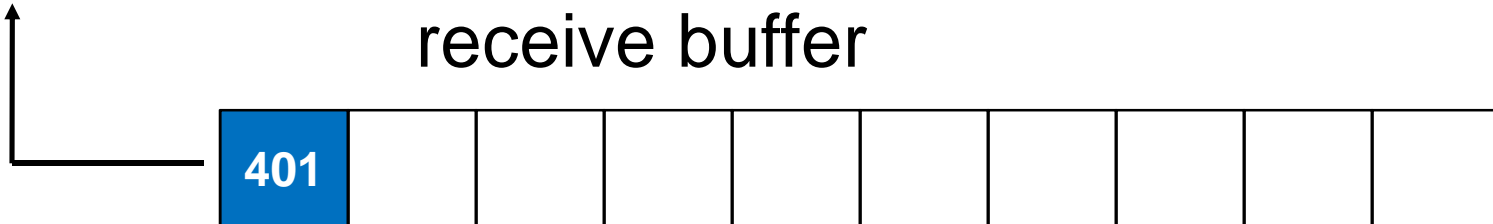
0000	00 13 72 0d 2b 14 00 16 3e 0d 23 37 08 00 45 00	...r.+...>.#7..E.
0010	00 3c 00 2c 40 00 80 06 be 5d 80 04 1e 1e 80 04	<.,@...].....
0020	1e 0c 00 50 2a f8 a3 7b 7e 6d ab bc e9 32 a0 10	...P*..{ ~m...2..
0030	ff fe e7 82 00 00 01 01 05 12 ab bc e9 39 ab bc9..
0040	e9 3a ab bc e9 37 ab bc e9 387..8

MISBEHAVIOR
B

Misbehavior B: filling gap between Cum ACK and first SACK

receiving
application

receive buffer



ordered data (ACKed)



out-of-order data (SACKed)

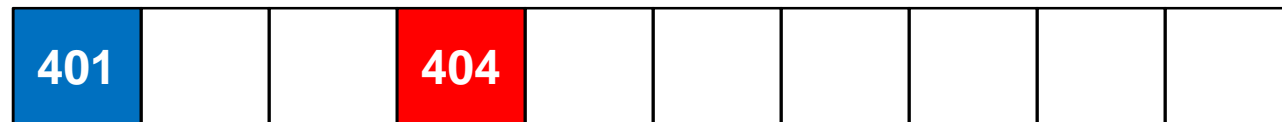


available space

Misbehavior B: filling gap between Cum ACK and first SACK

receiving
application

receive buffer



ordered data (ACKed)



out-of-order data (SACKed)

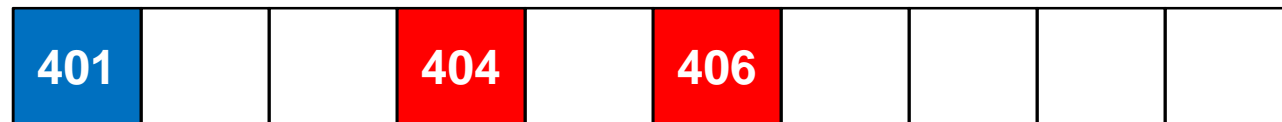


available space

Misbehavior B: filling gap between Cum ACK and first SACK

receiving
application

receive buffer



ordered data (ACKed)



out-of-order data (SACKed)

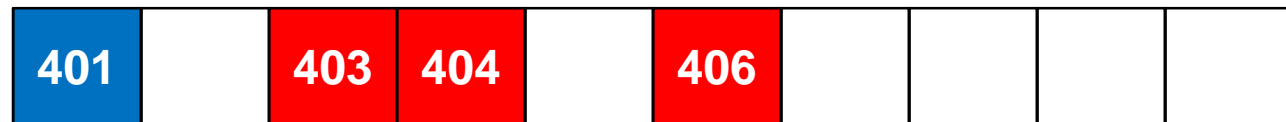


available space

Misbehavior B: filling gap between Cum ACK and first SACK

receiving
application

receive buffer



ordered data (ACKed)



out-of-order data (SACKed)

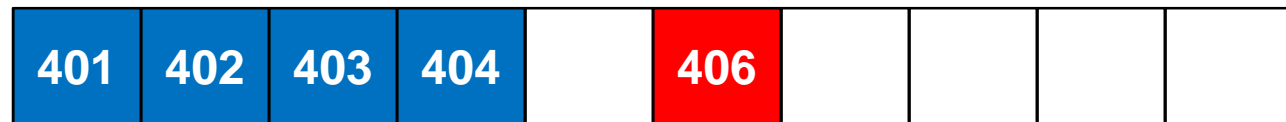


available space

Misbehavior B: filling gap between Cum ACK and first SACK

receiving
application

receive buffer



ordered data (ACKed)

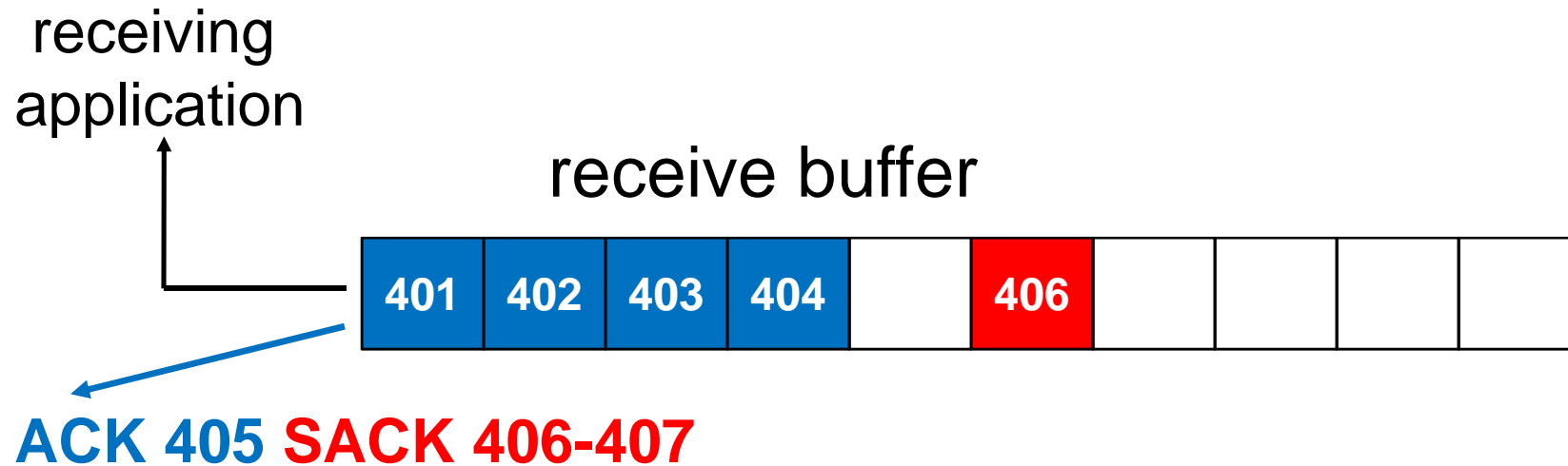





out-of-order data (SACKed)



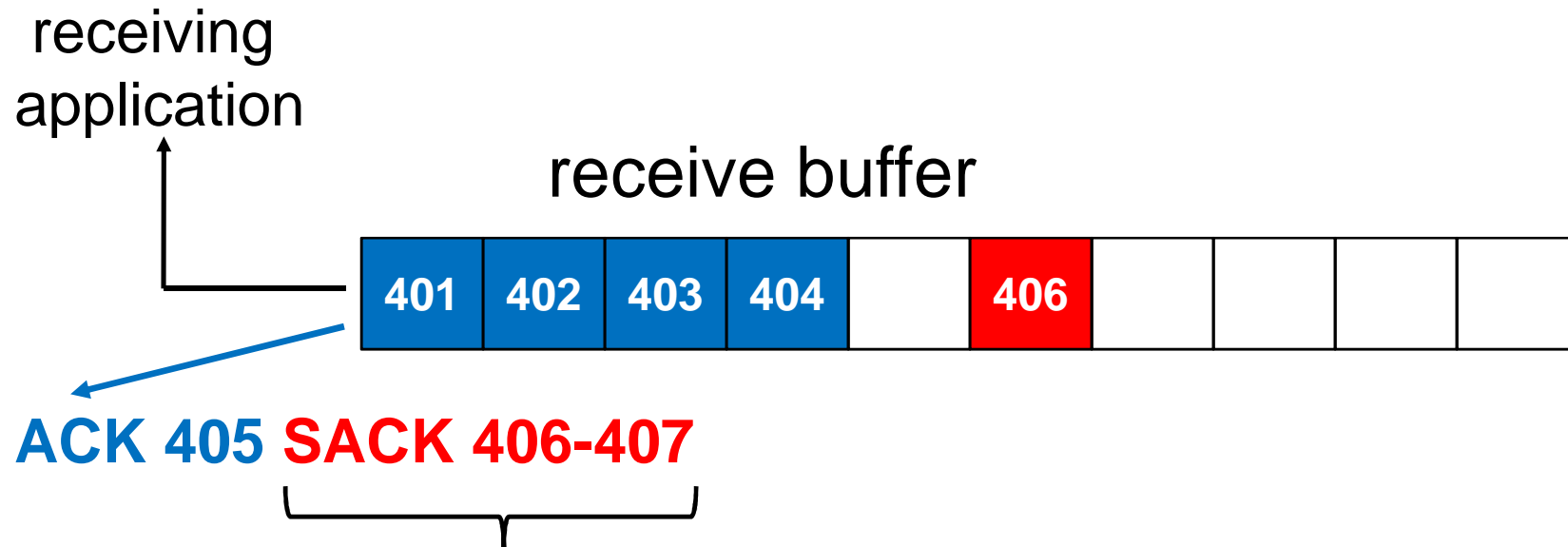
available space

Misbehavior B: filling gap between Cum ACK and first SACK






-  ordered data (ACKed)
-  out-of-order data (SACKed)
-  available space

Misbehavior B: filling gap between Cum ACK and first SACK



Misbehavior B: **missing**

-  ordered data (ACKed)
-  out-of-order data (SACKed)
-  available space

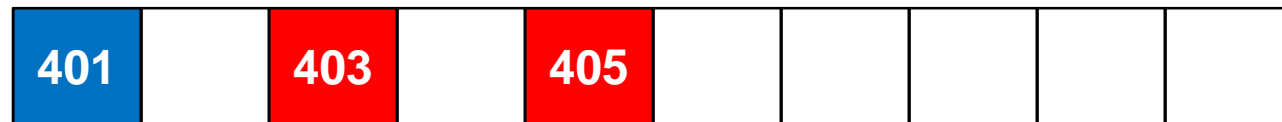
**MISBEHAVIOR
C**

Misbehavior C:

filling gap between two previously reported SACKs

receiving
application

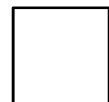
receive buffer



ordered data (ACKed)



out-of-order data (SACKed)



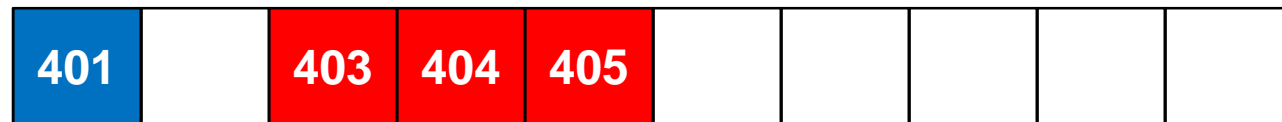
available space

Misbehavior C:

filling gap between two previously reported SACKs

receiving
application

receive buffer



ordered data (ACKed)



out-of-order data (SACKed)



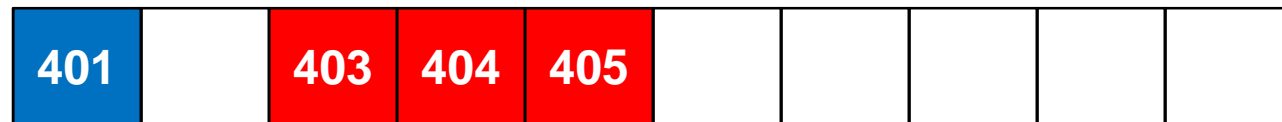
available space

Misbehavior C:

filling gap between two previously reported SACKs

receiving
application

receive buffer



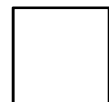
ACK 402 **SACK 403-406**



ordered data (ACKed)



out-of-order data (SACKed)



available space

Misbehavior C:

filling gap between two previously reported SACKs



ACK 402 ~~**SACK 403-406**~~
ACK 402 **SACK 403-405**

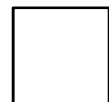
Misbehavior C



ordered data (ACKed)



out-of-order data (SACKed)

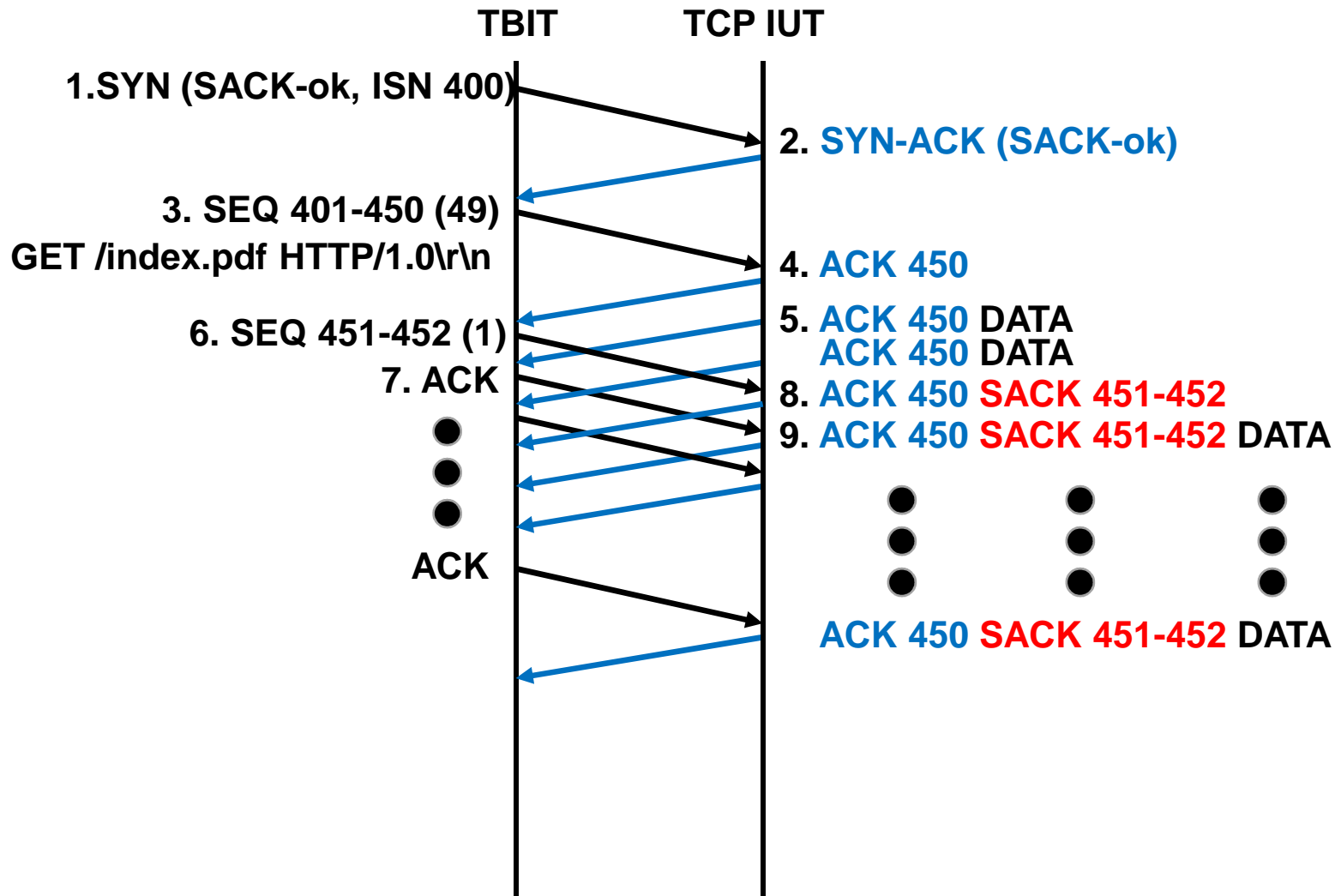


available space

**MISBEHAVIOR
D**

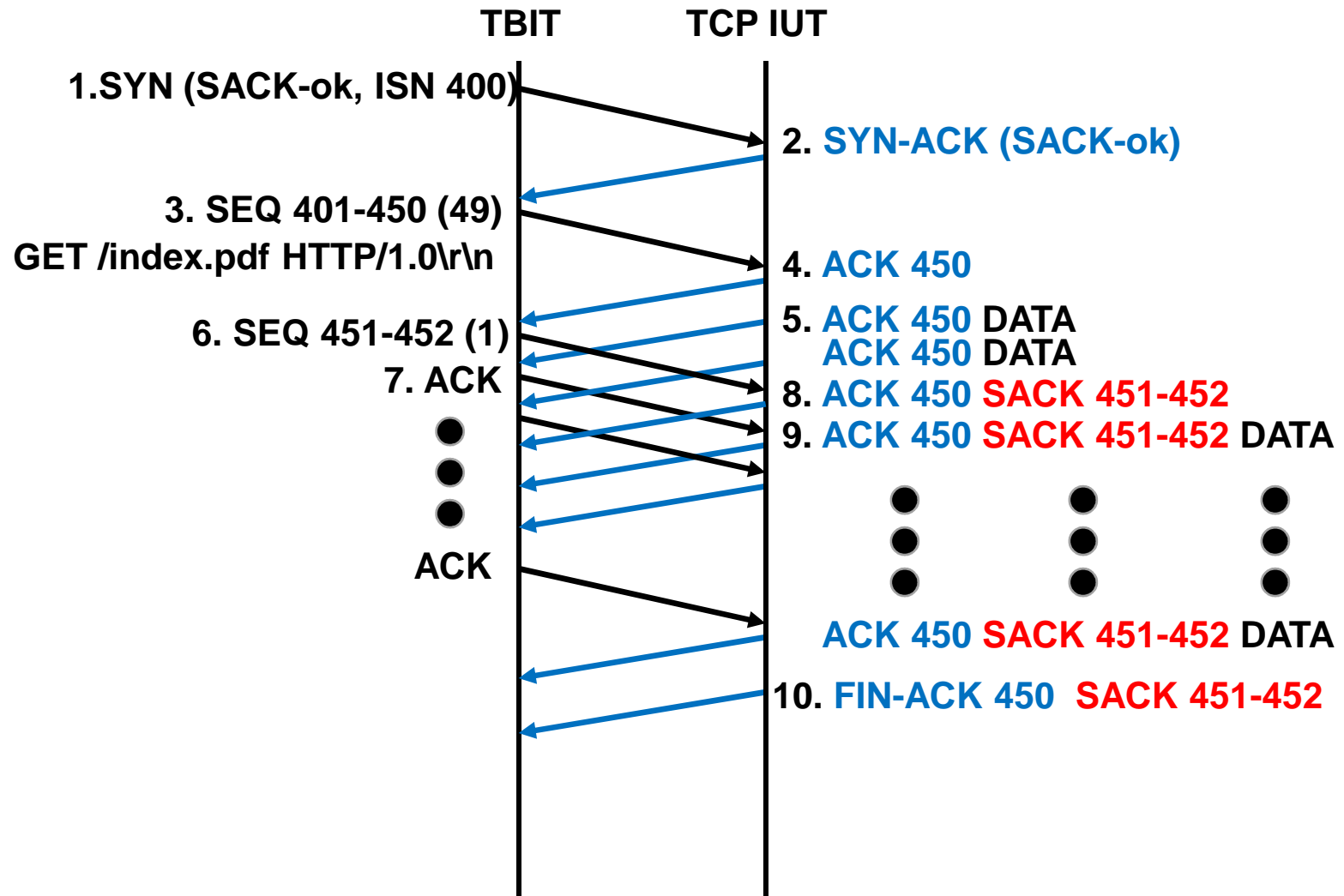
Misbehavior D:

failure to report SACKs in FIN segments

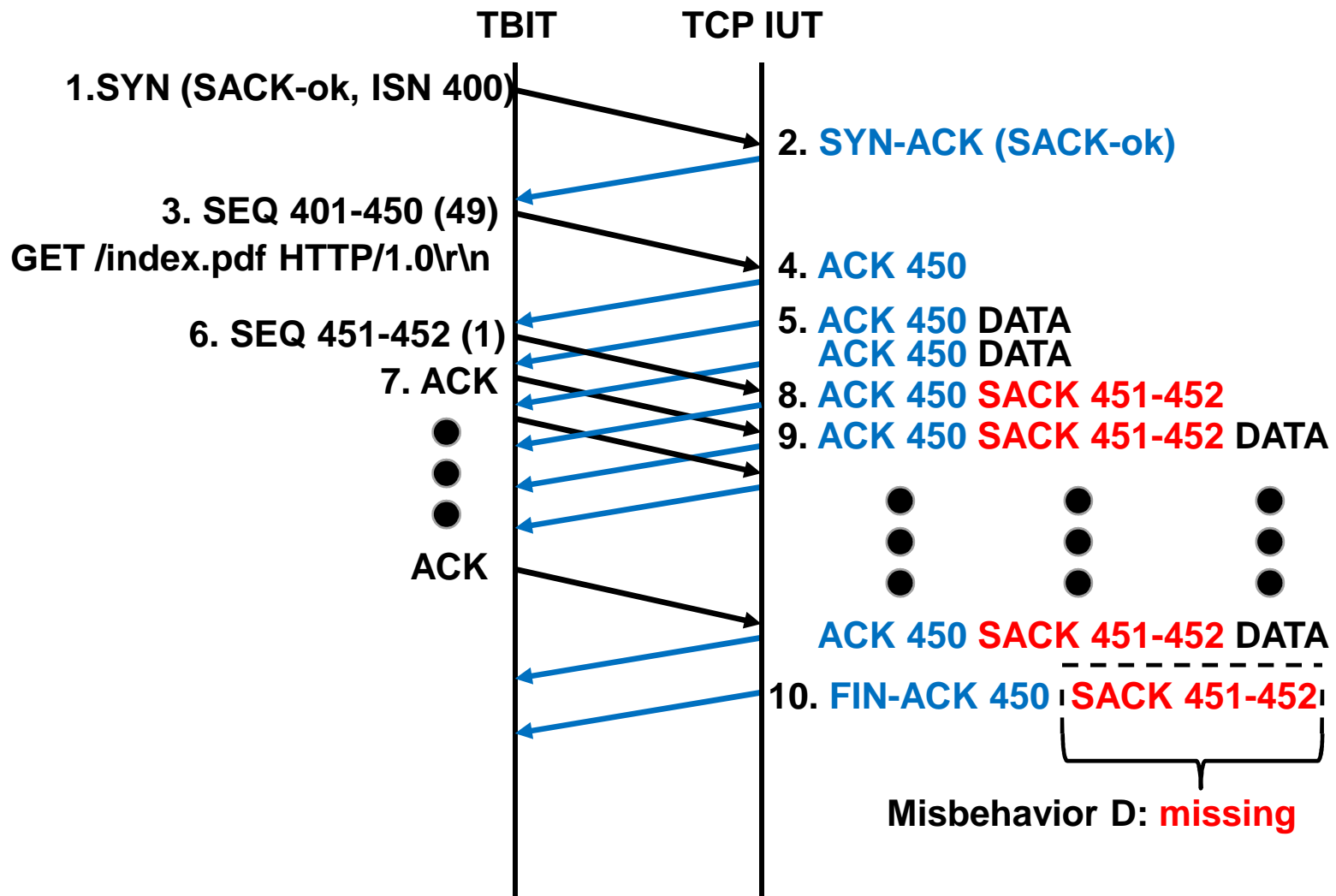


Misbehavior D:

failure to report SACKs in FIN segments



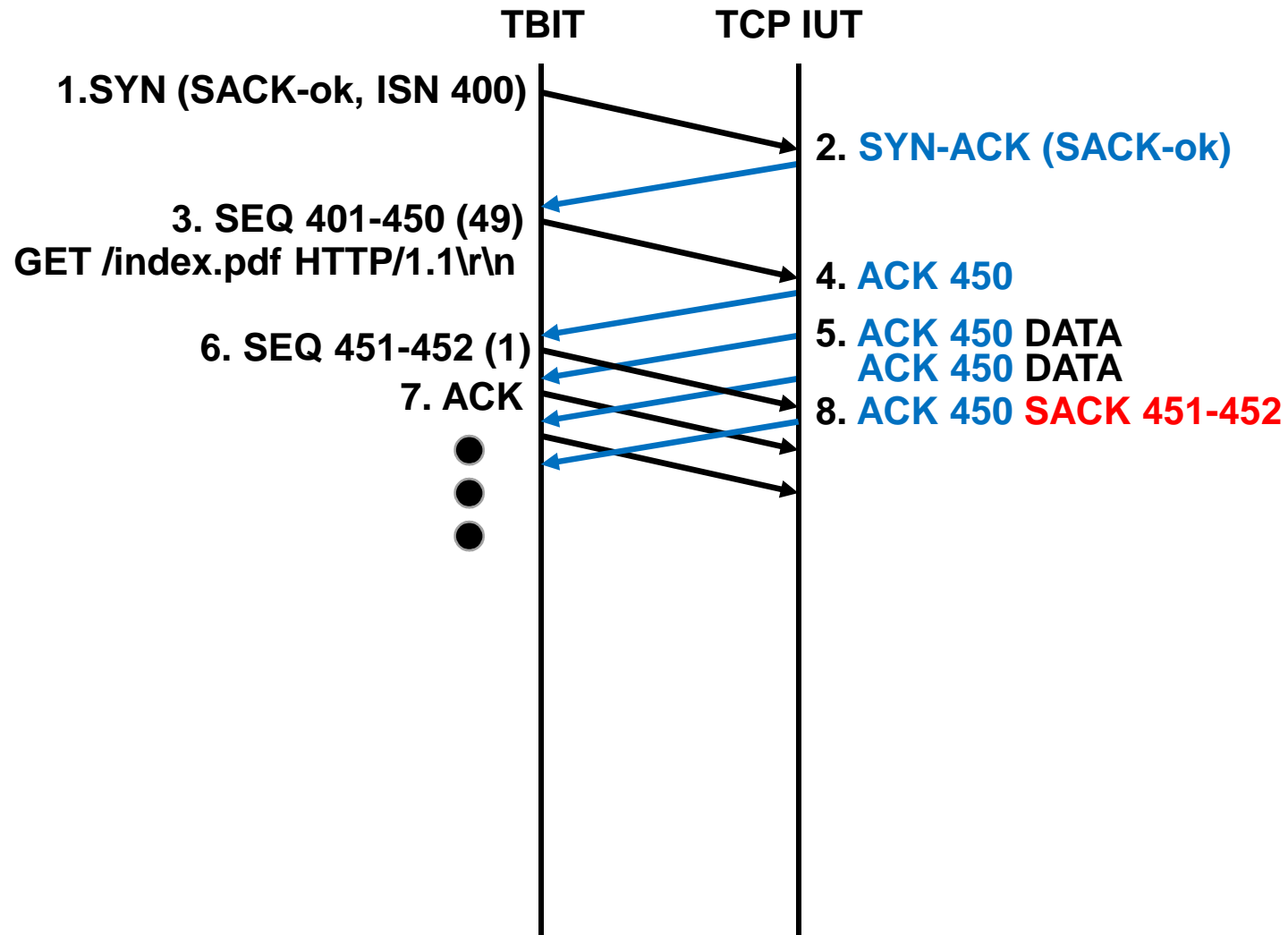
Misbehavior D: failure to report SACKs in FIN segments



**MISBEHAVIOR
E**

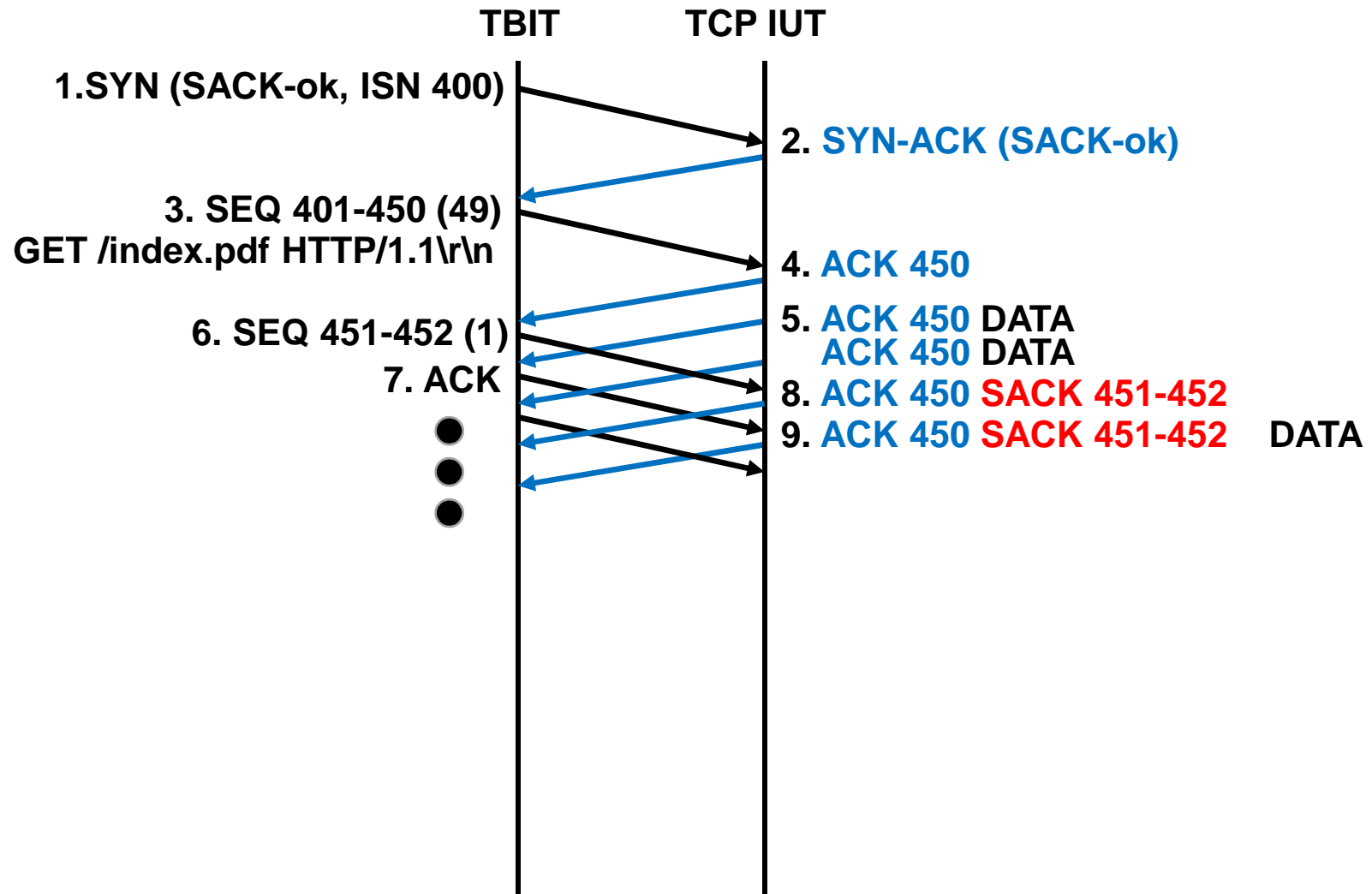
Misbehavior E:

failure to report SACKs in bidirectional data flow



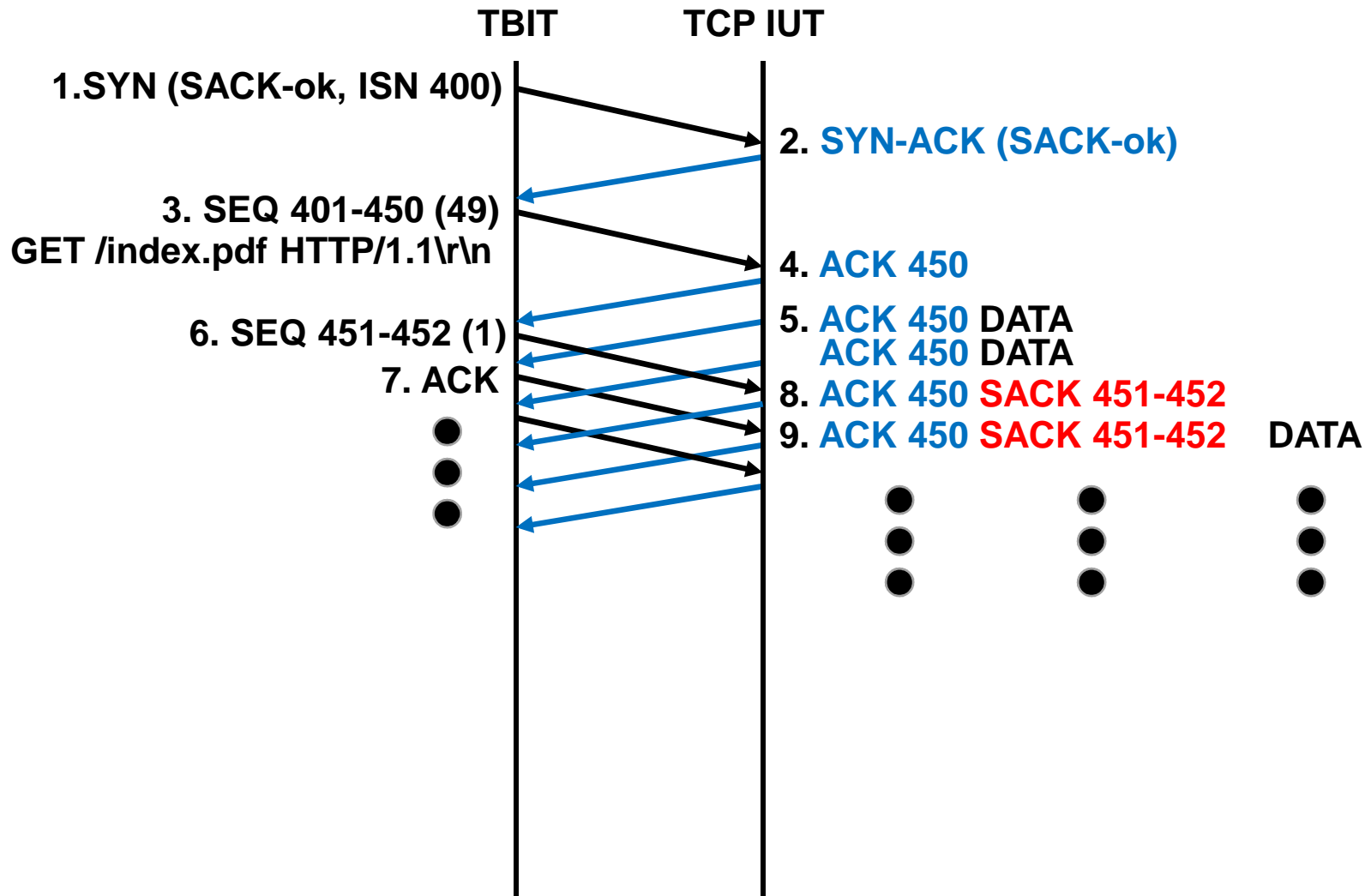
Misbehavior E:

failure to report SACKs in bidirectional data flow



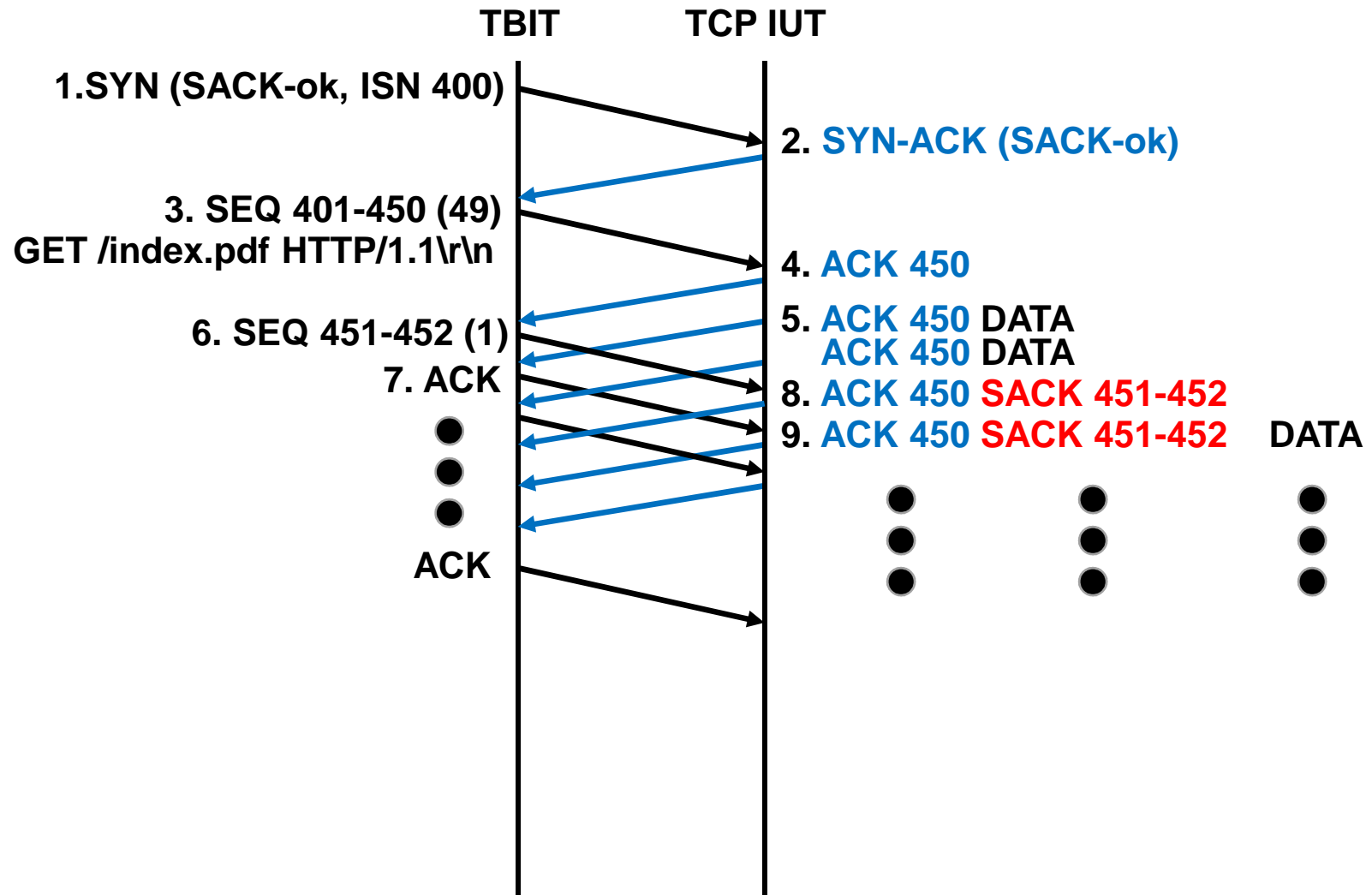
Misbehavior E:

failure to report SACKs in bidirectional data flow



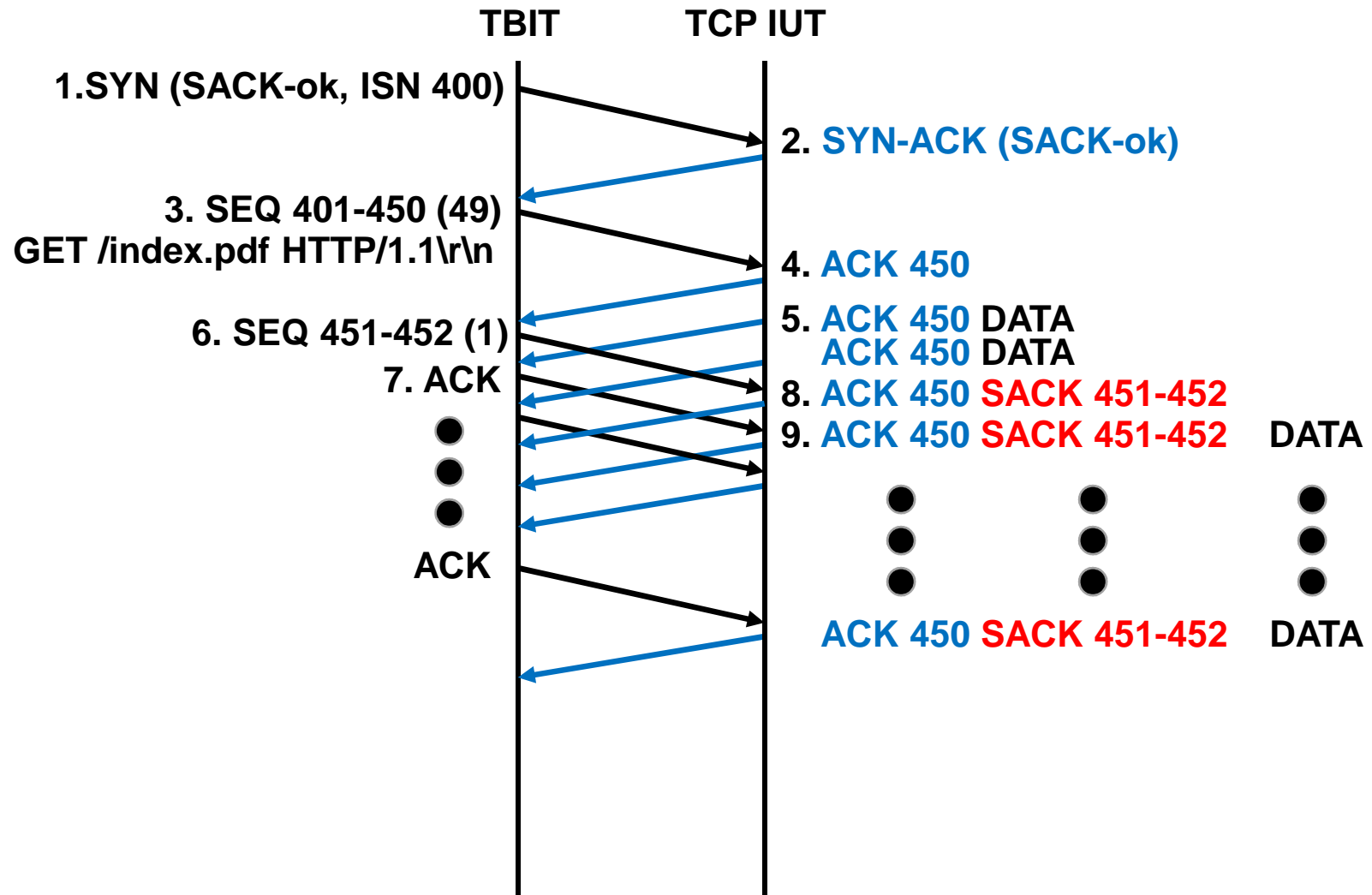
Misbehavior E:

failure to report SACKs in bidirectional data flow



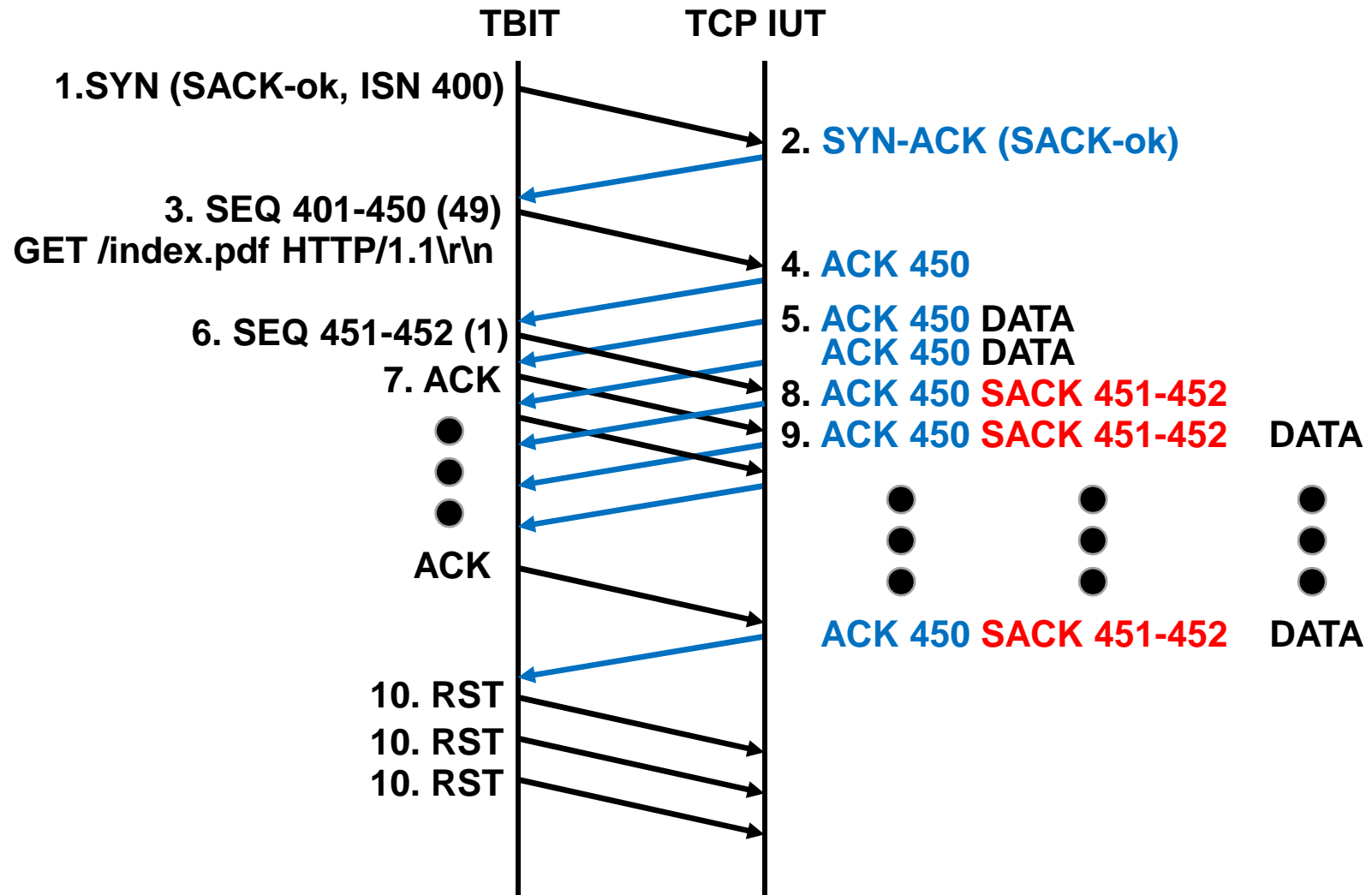
Misbehavior E:

failure to report SACKs in bidirectional data flow



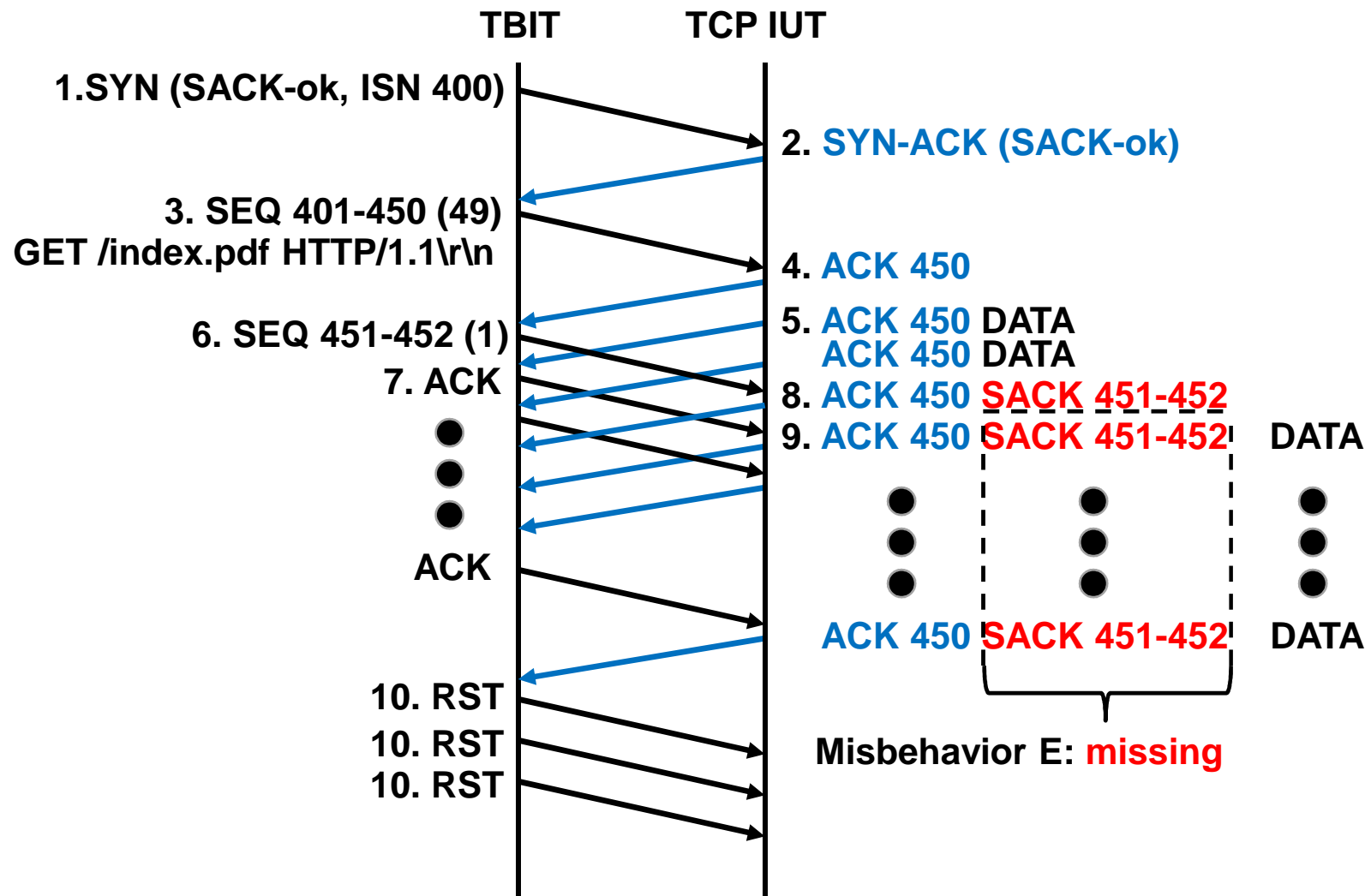
Misbehavior E:

failure to report SACKs in bidirectional data flow



Misbehavior E:

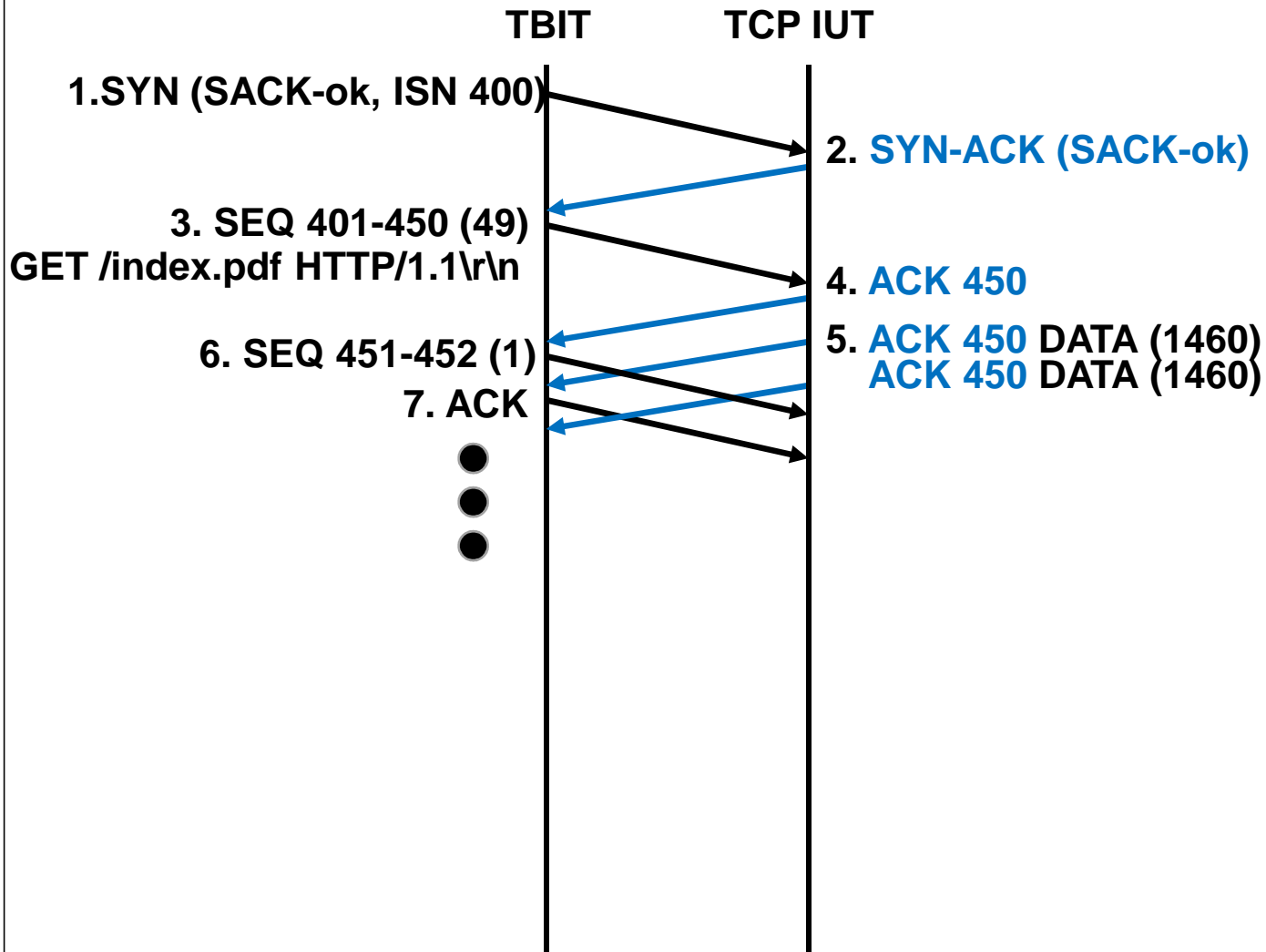
failure to report SACKs in bidirectional data flow



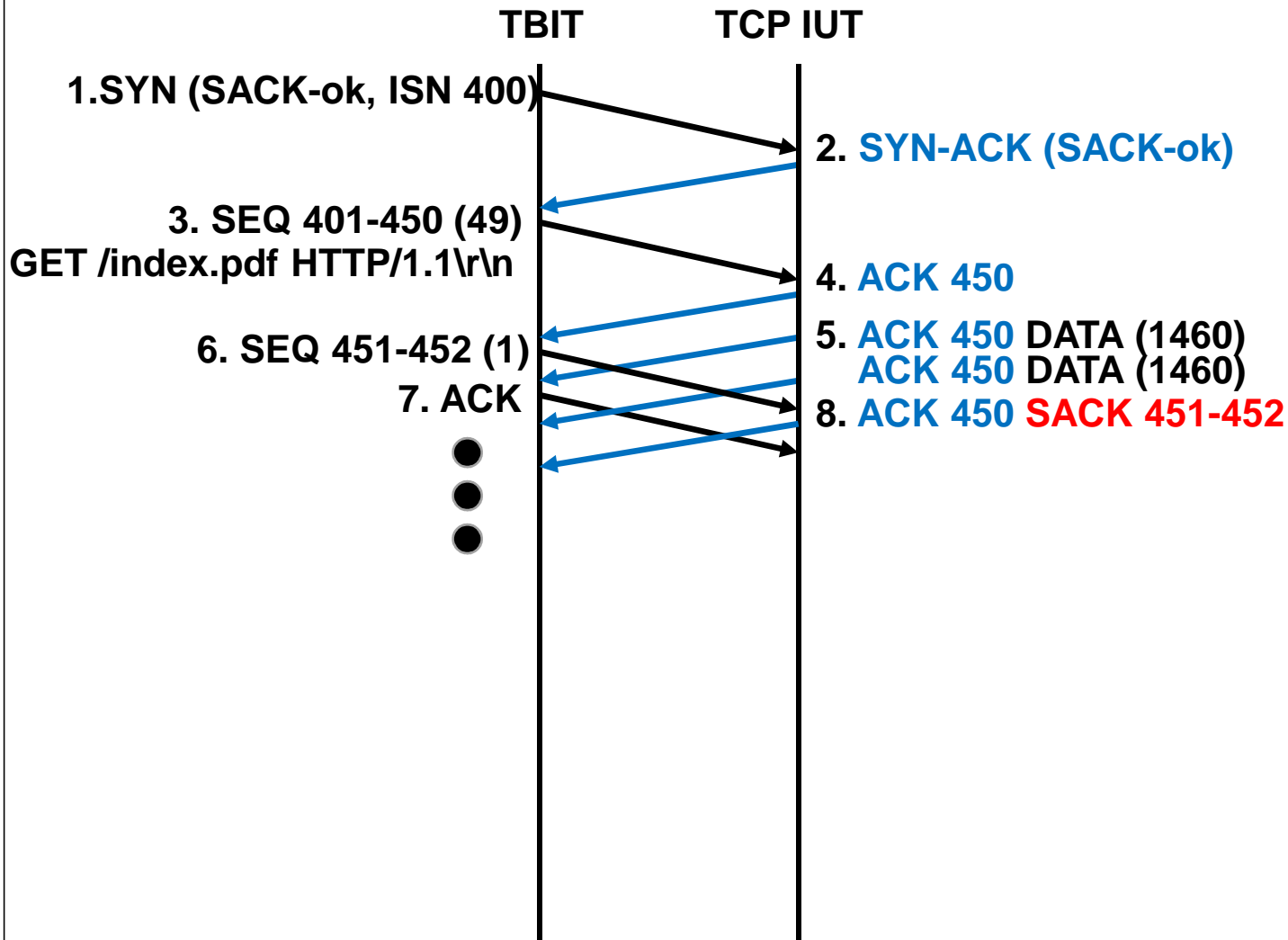
MISBEHAVIOR

F

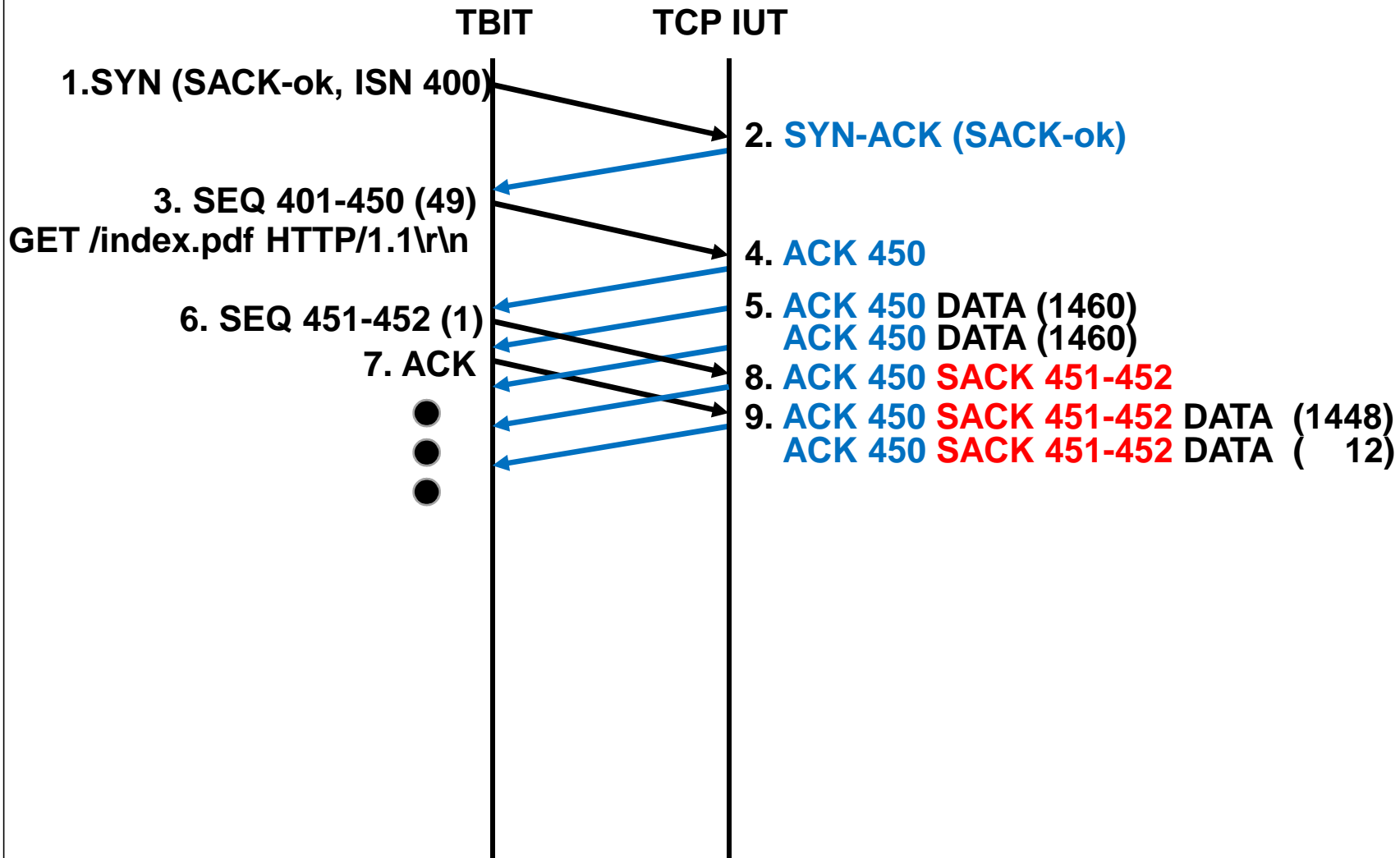
Misbehavior F: mishandling of data due to SACK processing



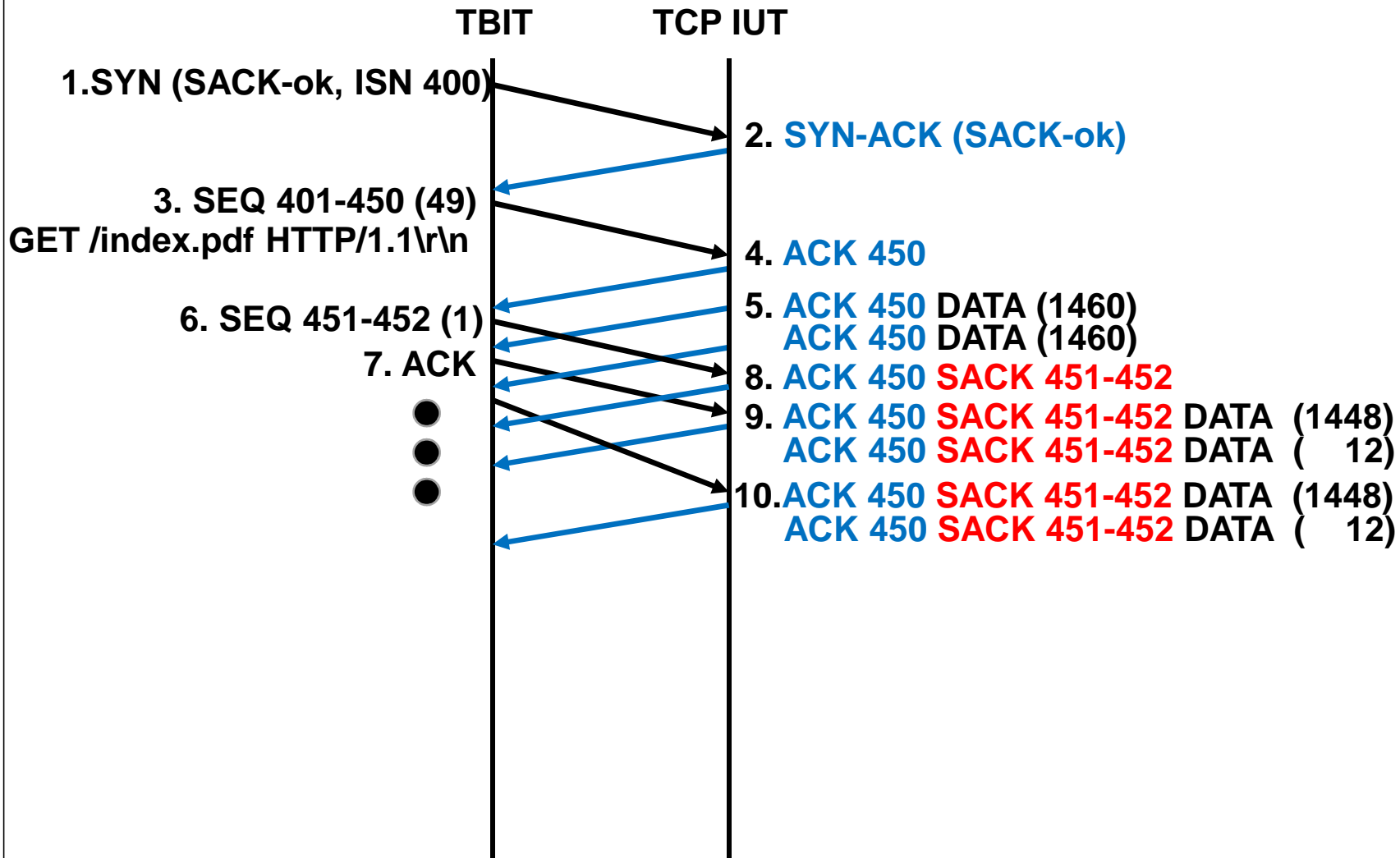
Misbehavior F: mishandling of data due to SACK processing



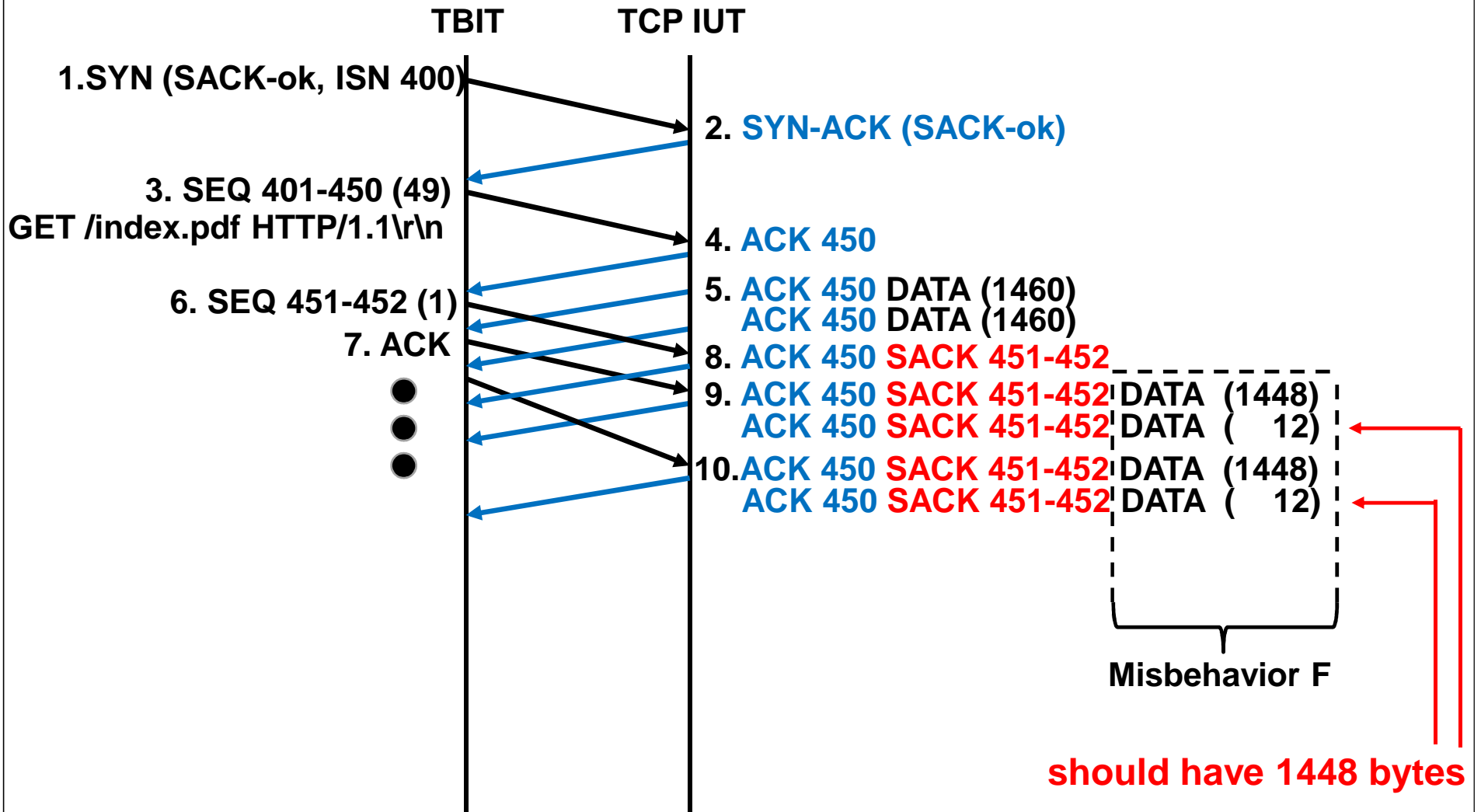
Misbehavior F: mishandling of data due to SACK processing



Misbehavior F: mishandling of data due to SACK processing



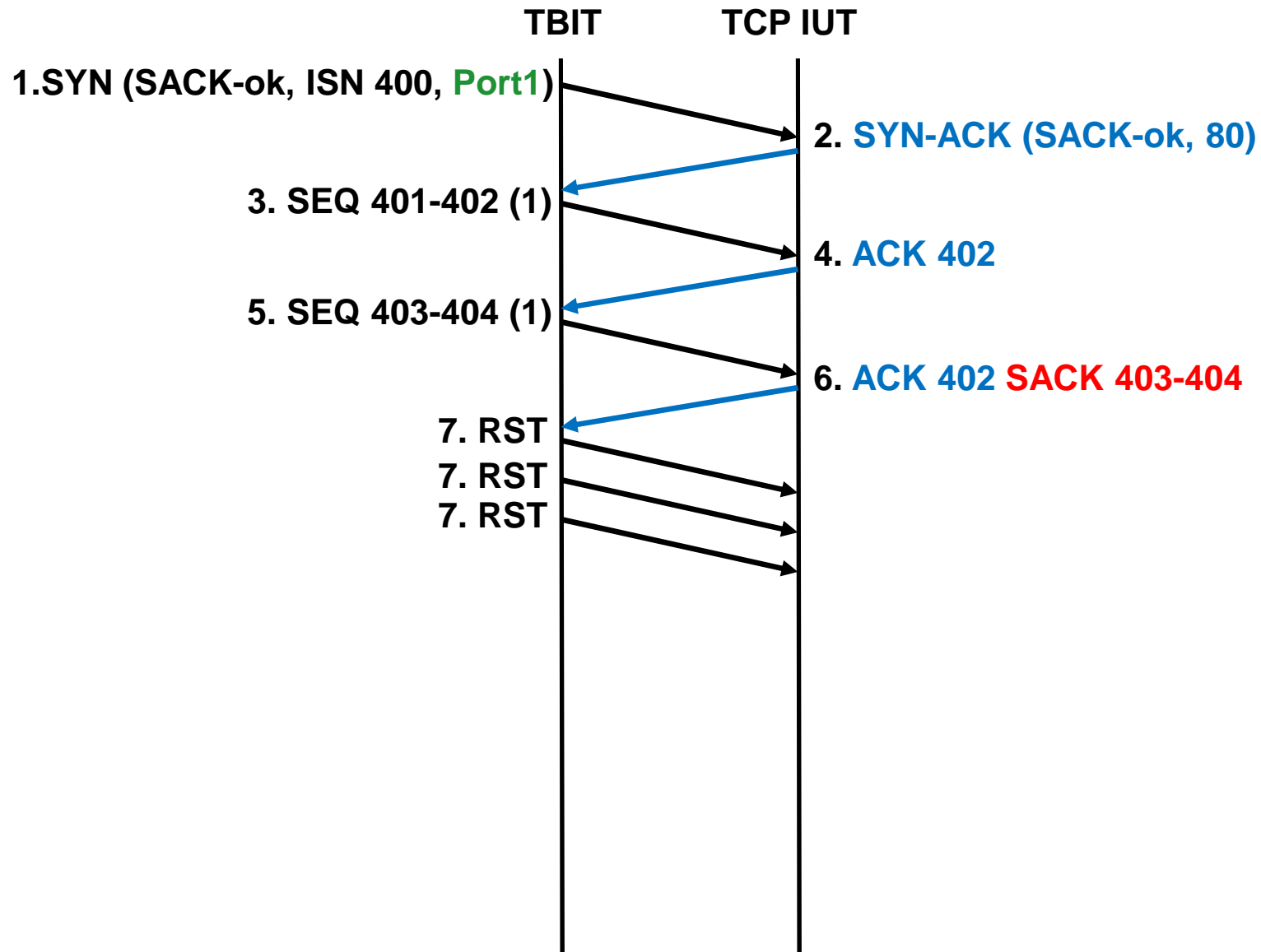
Misbehavior F: mishandling of data due to SACK processing



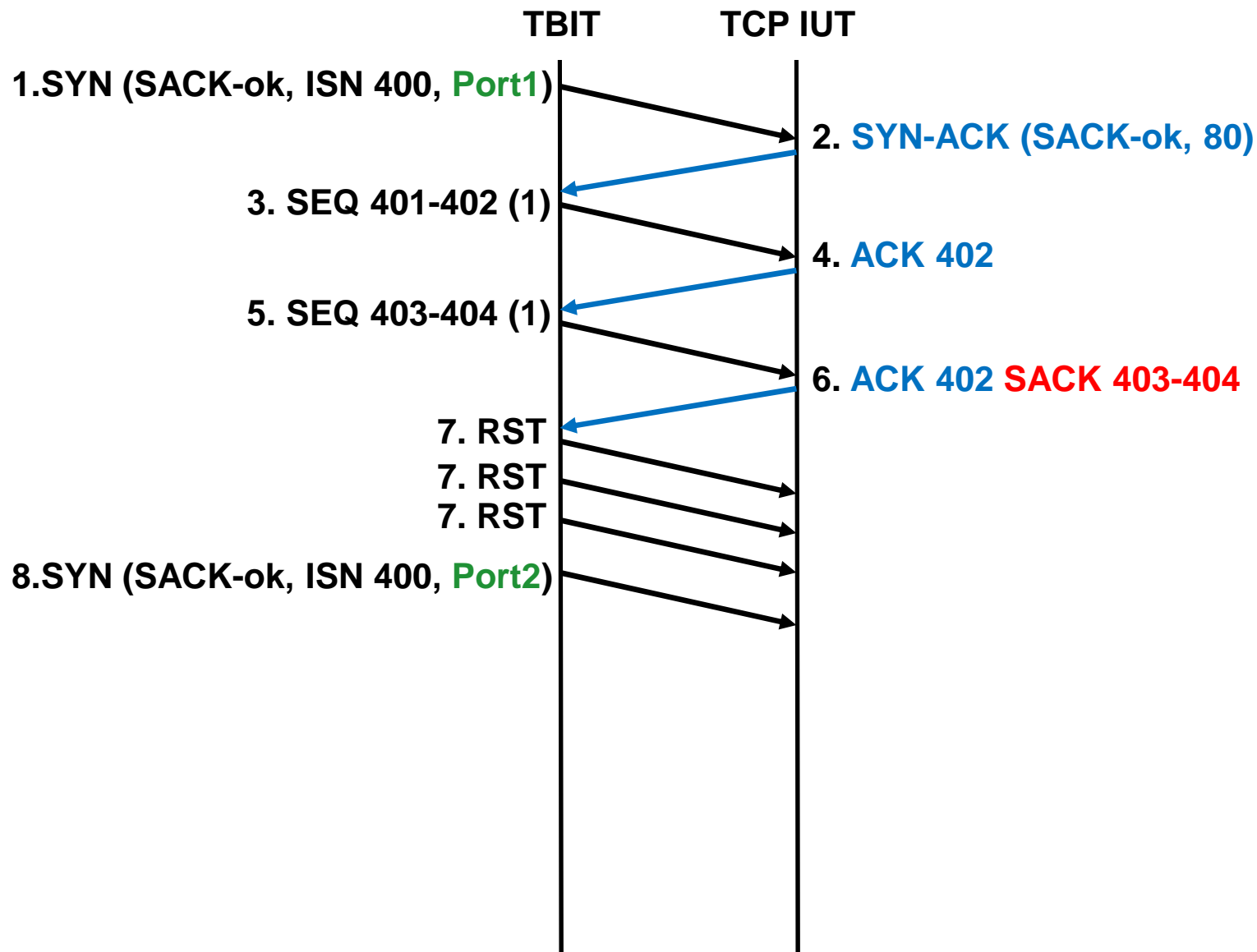
MISBEHAVIOR

G

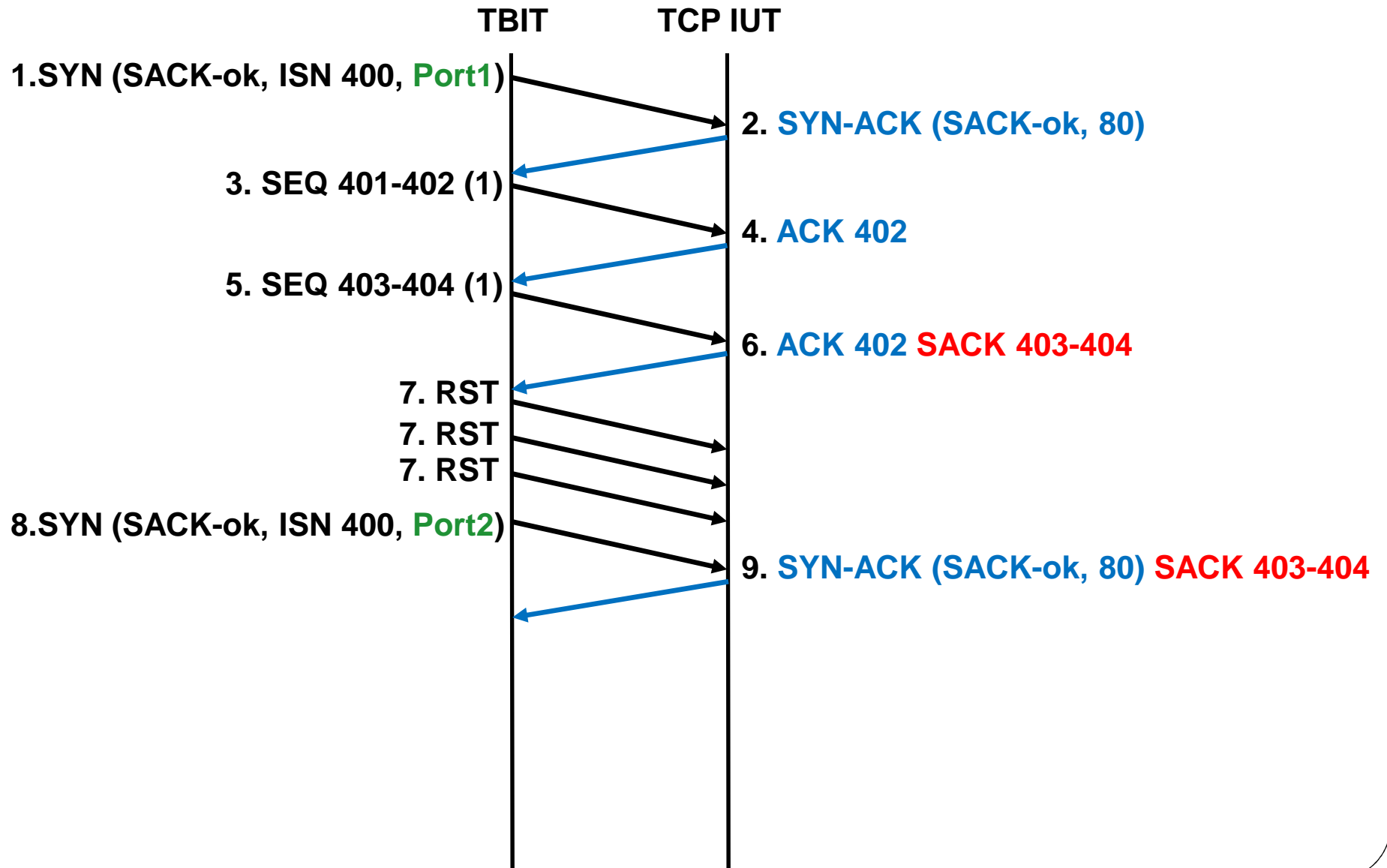
Misbehavior G: SACK reappearance in consecutive connections



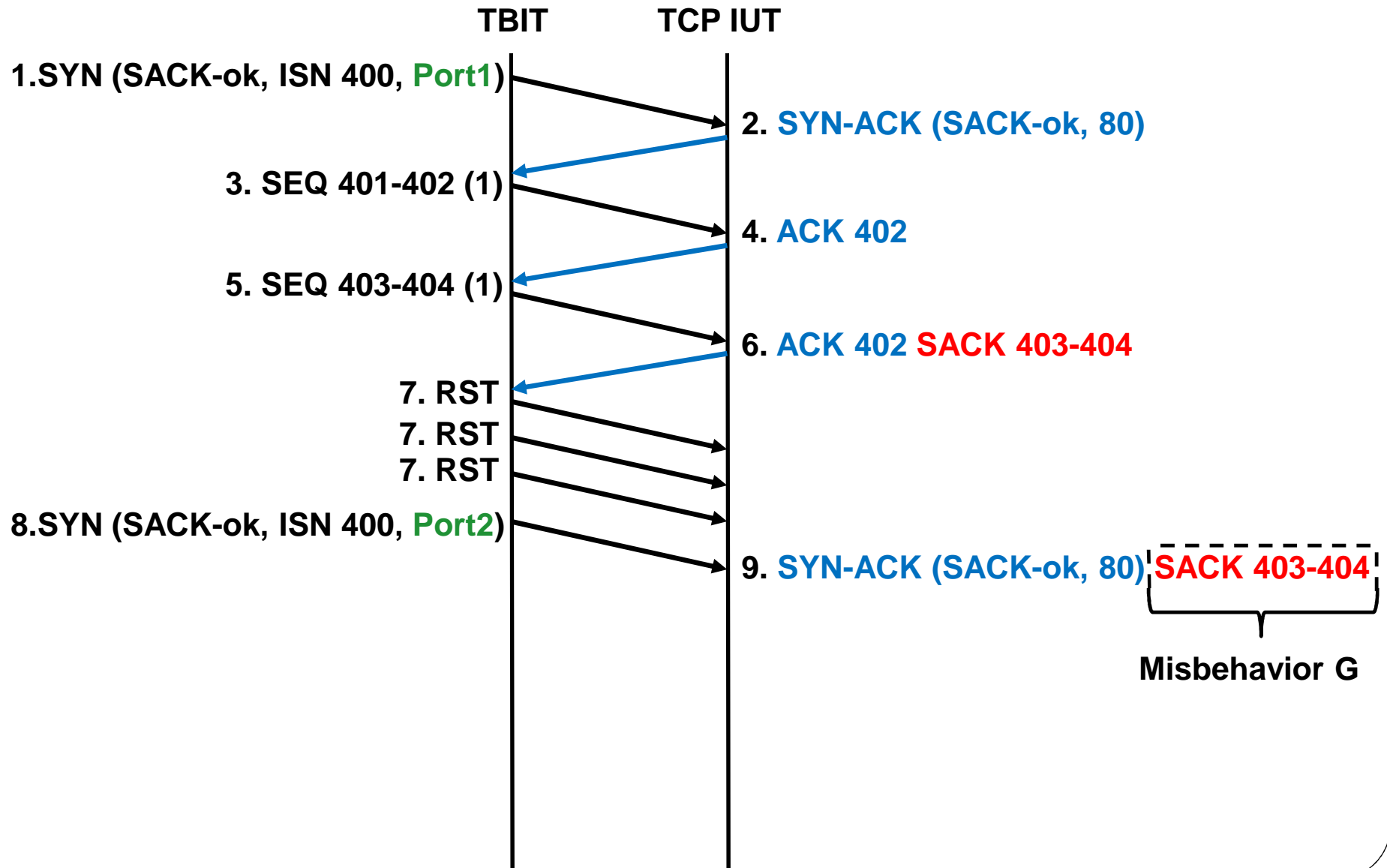
Misbehavior G: SACK reappearance in consecutive connections



Misbehavior G: SACK reappearance in consecutive connections



Misbehavior G: SACK reappearance in consecutive connections



OUTLINE

1. Introduction
2. Experimental design
3. 7 SACK misbehaviors
4. Results
5. Summary

Results

Operating System	Misbehavior						
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>
FreeBSD 5.3, 5.4	Y			Y			
Linux 2.2.20 (Debian 3)						Y	
Linux 2.4.18 (Red Hat 8)						Y	
Linux 2.4.22 (Fedora 1)						Y	
Linux 2.6.12 (Ubuntu 5.10)						Y	
Linux 2.6.15 (Ubuntu 6.06)						Y	
Linux 2.6.18 (Debian 4)						Y	
OpenBSD 4.2, 4.5, 4.6, 4.7	Y			Y			
OpenSolaris 2008.05						Y	Y
OpenSolaris 2009.06						Y	Y
Solaris 10							Y
Solaris 11						Y	
Windows 2000	Y	Y	Y	Y	Y		
Windows XP	Y	Y	Y	Y	Y		
Windows Server 2003	Y	Y	Y	Y	Y		
Windows Vista				Y	Y		
Windows Server 2008				Y	Y		
Windows 7				Y	Y		

Results

Operating System	Misbehavior						
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>
FreeBSD 6.0							
FreeBSD 7.3							
FreeBSD 8.0							
Linux 2.6.31 (Ubuntu 9.10)							
Mac OS X 10.5							
Mac OS X 10.6							

OUTLINE

1. Introduction
2. Experimental design
3. 7 SACK misbehaviors
4. Results
5. Summary

Summary

- we accidentally discovered 7 unexpected SACK behaviors while researching renegeing
- we designed a methodology using TBIT to identify OSes exhibiting these misbehaviors
- we applied the methodology on 29 OSes
 - found at least one OS for each misbehavior
- Conclusion: SACKs - simple in concept; complex to implement

Nasif Ekiz, Abuthahir Rahman, Paul Amer

"Misbehaviors in TCP SACK Generation"

ACM Computer Communications Review, 41(2), 4/2011

Questions?