Optimizing TCP Forwarder Performance

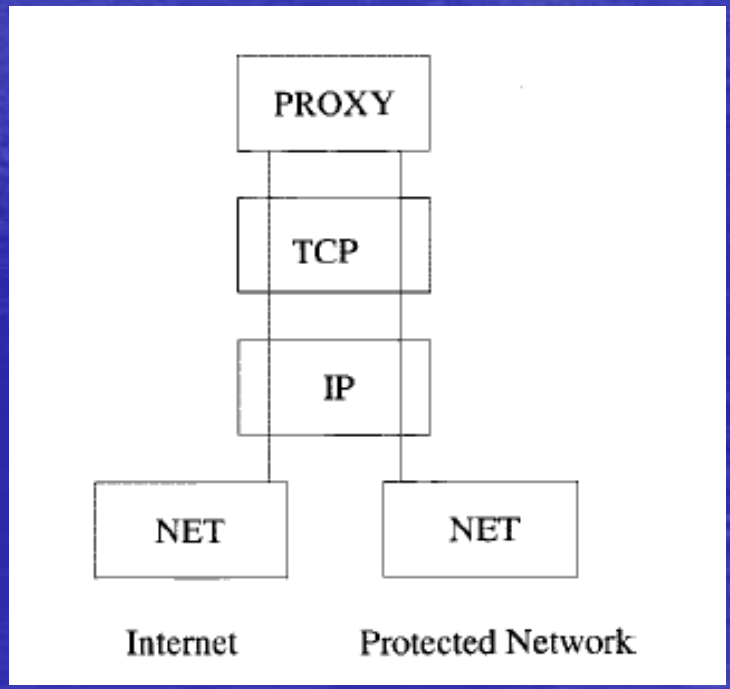
Oliver Spatscheck, Jørgen S. Hansen, Student Member, IEEE,
John H. Hartman, Member, IEEE, and
Larry L. Peterson, Senior Member, IEEE

Outline

- TCP Forwarder
- Connection splicing
- Connection scout
- Test
- Conclusion

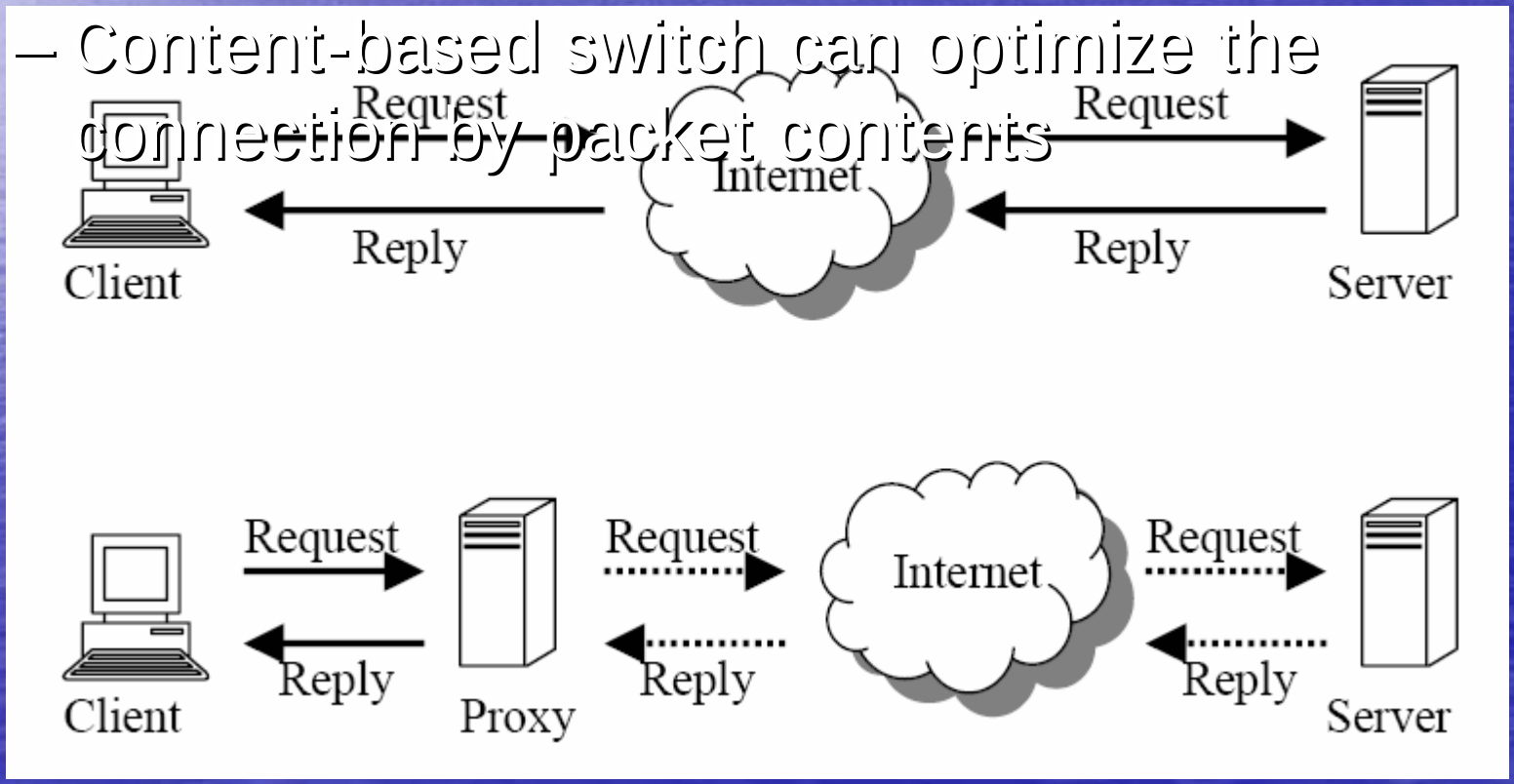
TCP Forwarder

- Firewall
  - ex: when accept connections , need authentication



TCP Forwarder

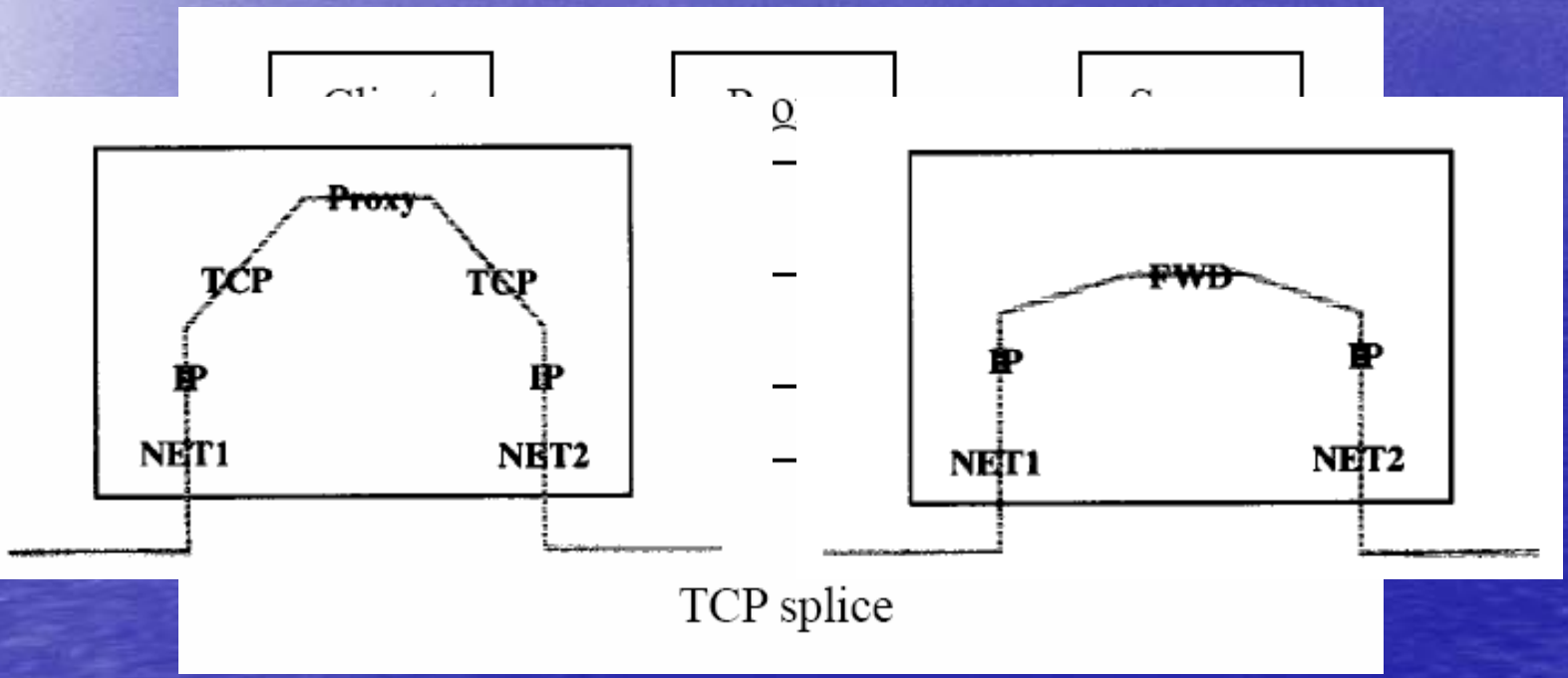
- HTTP proxy
  - Content-based switch can optimize the connection by packet contents



Connection splicing

- Proxy mode
  - Control mode : firewall authentication
  - Forward mode : HTTP proxy forward packet from server
- Forward mode
  - It only adjust headers or cache data
  - Where is overhead ?

Connection splicing

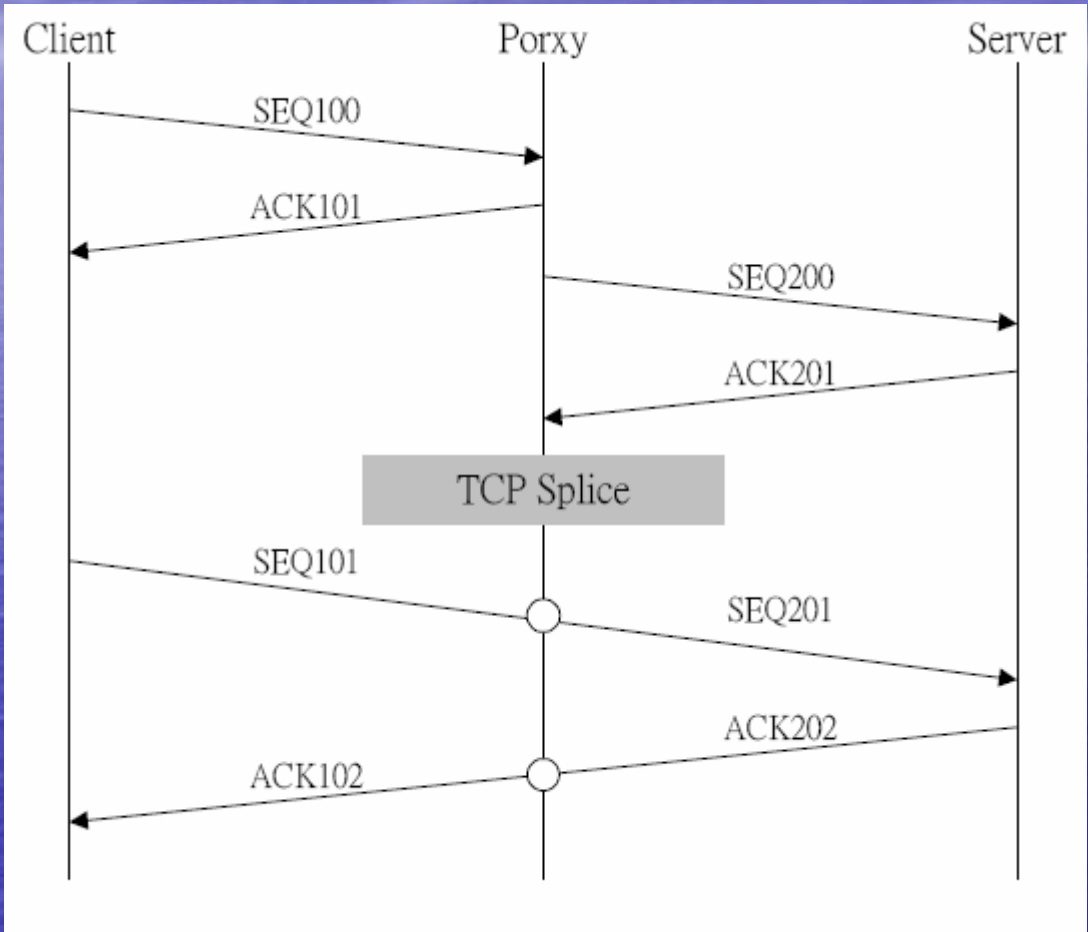


TCP splice

Connection splicing

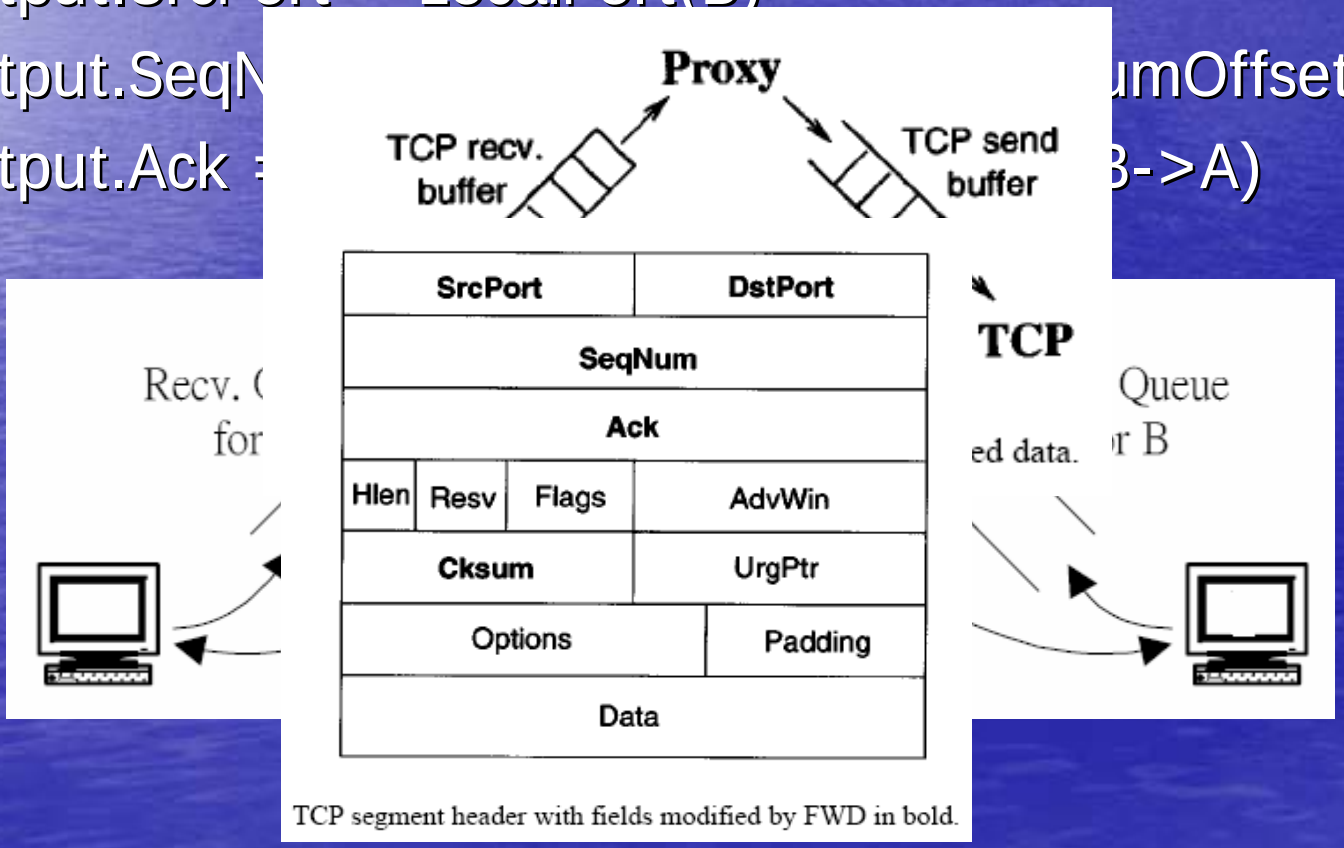
- Forwarding
  - Port number : two TCP connection may be other
  - Seq. number : it will be different, because TCP initialize
  - ACK number : it is dissimilar by the direction
  - Checksum : modify header have to adjust

Connection splicing



Connection splicing

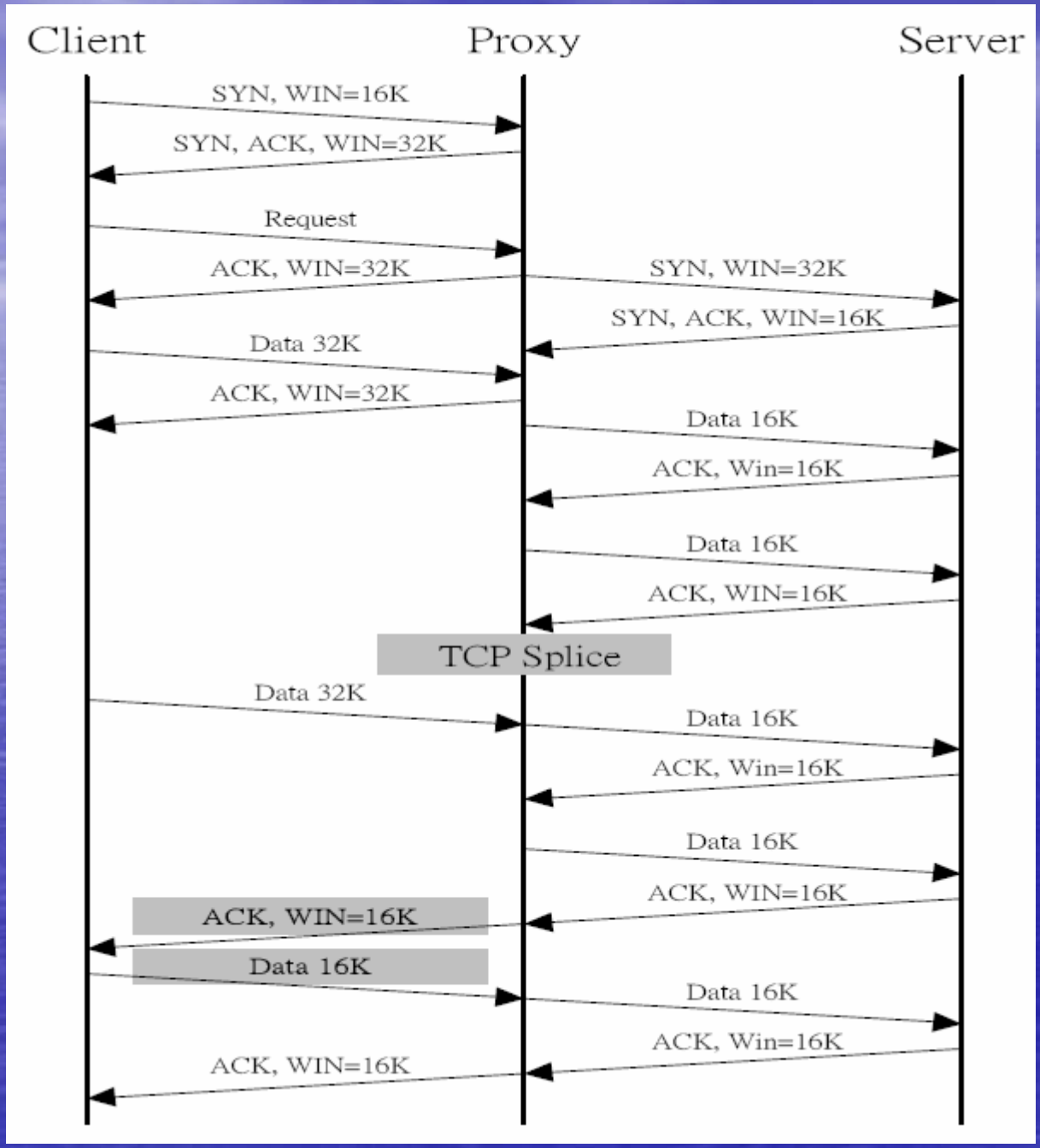
- Output.DstPort = RemotePort(B)
- Output.SrcPort = LocalPort(B)
- Output.SeqN umOffset(A->B)
- Output.Ack = B->A)

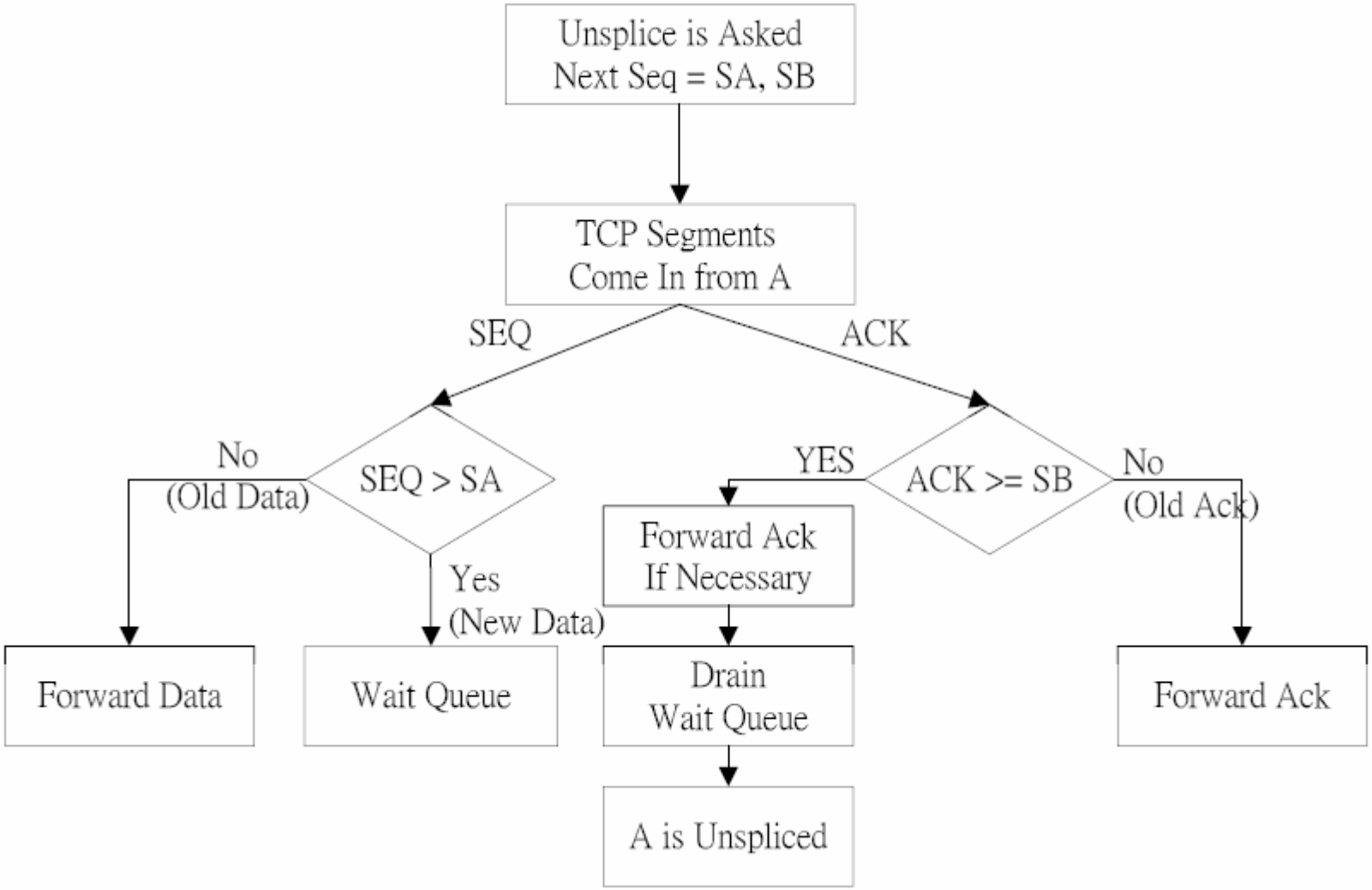


TCP segment header with fields modified by FWD in bold.

Connection splicing

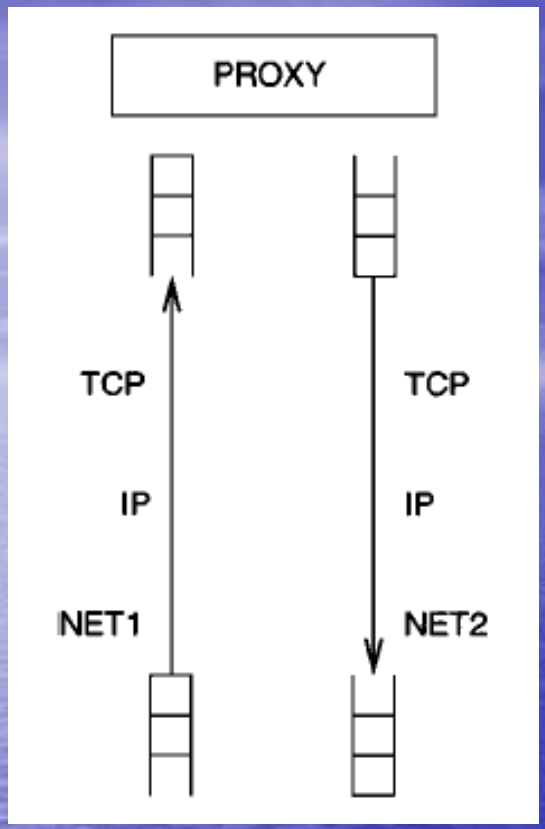
- When Splice, we should check the receive queue in unoptimized TCP
  - New packet put in wait queue
  - Sometimes have window size problem
- When unsplice
  - Check all packet are acknowledged in optimized TCP
  - Forward mode to control mode

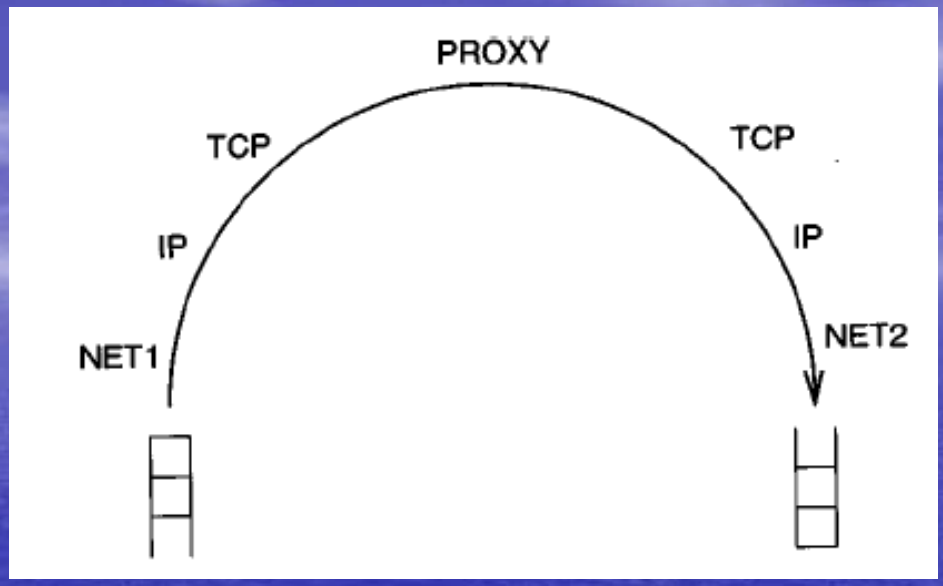

Client
Proxy
Server
SYN, WIN=16K
SYN, ACK, WIN=32K
Request
ACK, WIN=32K
SYN, WIN=32K
SYN, ACK, WIN=16K
Data 32K
ACK, WIN=32K
Data 16K
ACK, Win=16K
Data 16K
ACK, WIN=16K
TCP Splice
Data 32K
Data 16K
ACK, Win=16K
Data 16K
ACK, WIN=16K
ACK, WIN=16K
Data 16K
Data 16K
ACK, Win=16K
ACK, WIN=16K

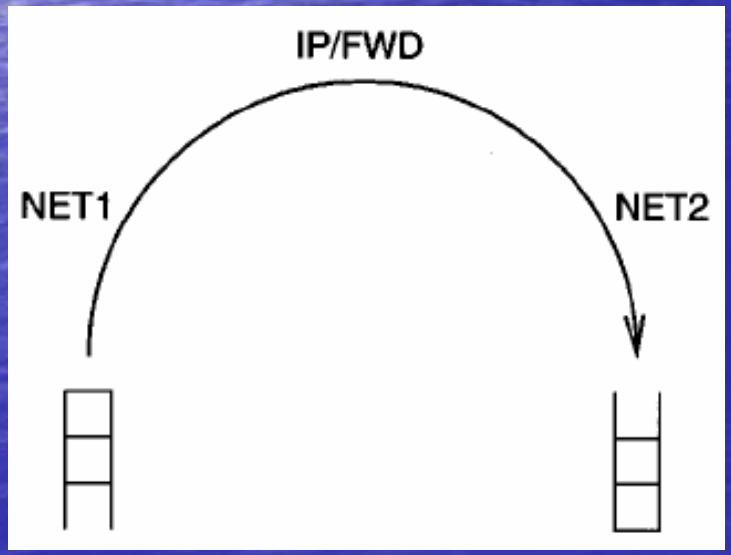

Unsplice is Asked
Next Seq = SA, SB
TCP Segments
Come In from A
SEQ
ACK
No
(Old Data)
SEQ > SA
Yes
(New Data)
YES
ACK >= SB
No
(Old Ack)
Forward Ack
If Necessary
Forward Data
Wait Queue
Drain
Wait Queue
Forward Ack
A is Unspliced

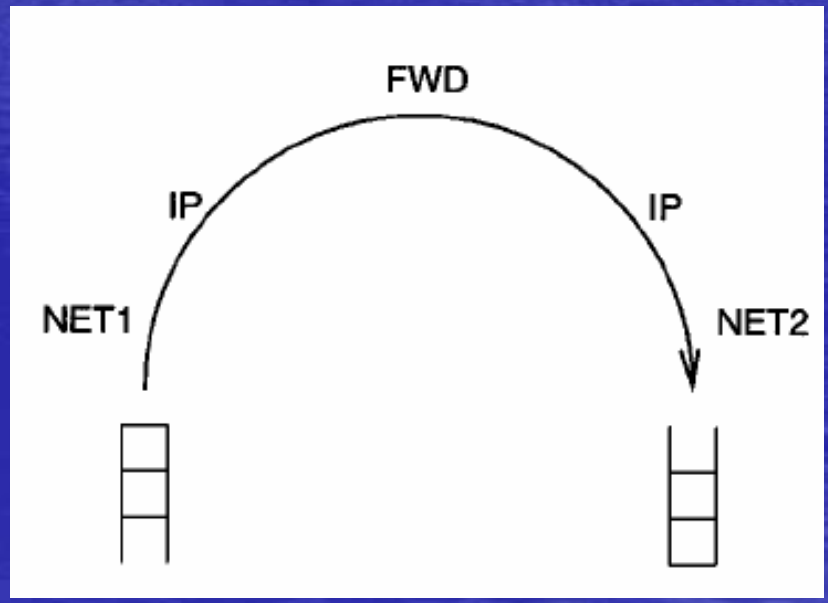
Connection scout

- Scout define path abstraction that encapsulates data
- Each path have two important define
  - A sequence of code module are applied to data
  - Represent the entity is scheduled for execution


PROXY
TCP
IP
NET1
TCP
IP
NET2

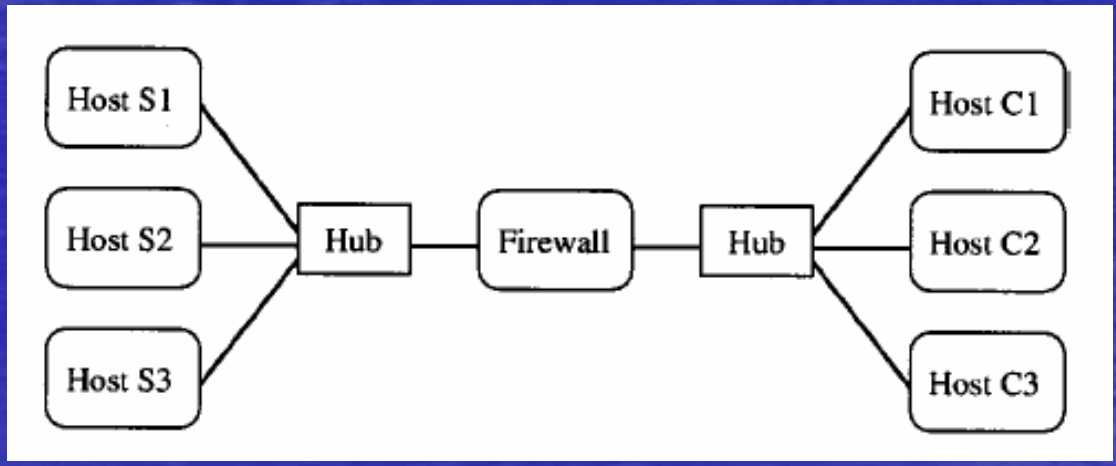

PROXY
TCP
IP
NET1
TCP
IP
NET2


IP/FWD
NET1
NET2

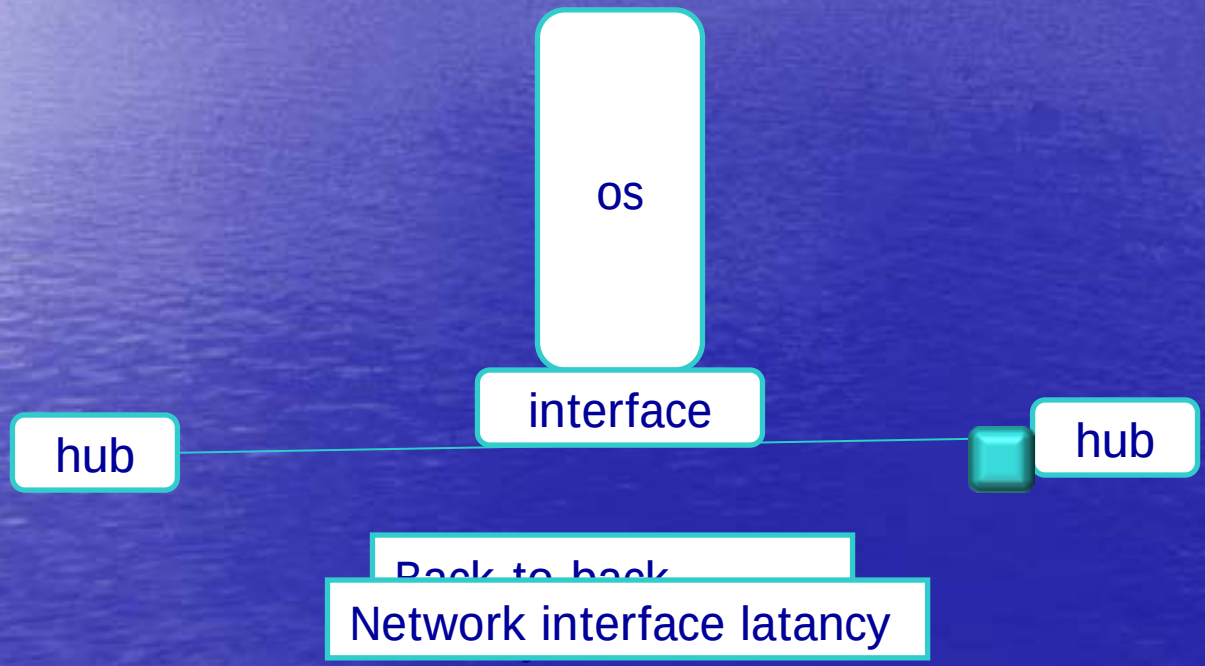

FWD
IP
NET1
IP
NET2

Test

- All host are 200MHz CPU with 256KB cache
- 128MB ram
- Digital Fast EtherWORKS PCI 10/100
- Linux 2.0.30



Test



Test

FIREWALL AND ROUTER PROCESSING PER TCP SEGMENT

Configuration		1-byte TCP segments		1460-byte TCP segments	
		Processing time (μsecs)	Speedup	Processing time (μsecs)	Speedup
Scout	2-path	68.5	–	101.1	–
	1-path	66.1	1.04	98.6	1.03
	FWD	39.0	1.76	39.5	2.56
	IP/FWD	24.0	2.85	24.0	4.21
	IP router	22.4	3.06	22.4	4.51
Linux	TIS Firewall	83.9	–	113.0	–
	Filtering IP router	27.5	3.05	29.0	3.90
	IP router	25.5	3.29	25.4	4.45

Conclusion

- We describes connection splicing can be applied to TCP forwarder
- In the future, we are interest in SSL-secured HTTP connections