Minimal TCP/IP implementation with proxy support

Adam Dunkels
SICS

adam@sics.se

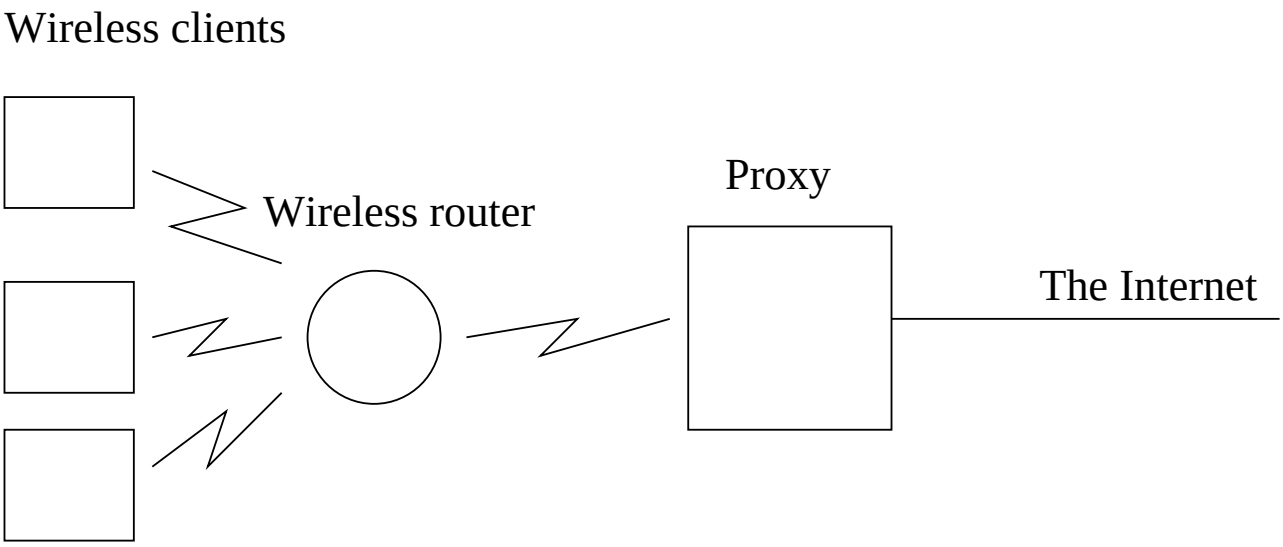
20 december 2000

Background

- Wireless networks with TCP/IP end-to-end
- Small devices — limited computing and memory resources
- A proxy can be used to offload the small devices

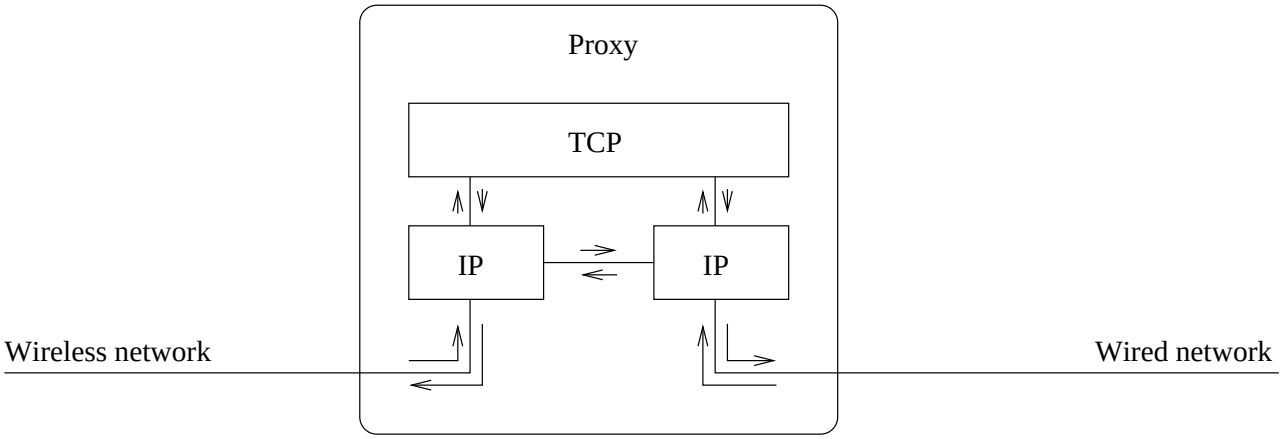
Goals

- Design/implementation of an offloading proxy
  - The proxy should not break the end-to-end principle
- Design/implementation of a small TCP/IP stack



- Example: Arena

The proxy



- Transparent
- Does not require changes to TCP/IP protocols
- Two mechanisms
  - Per packet processing (IP)
  - Per connection processing (TCP)
- Soft-state

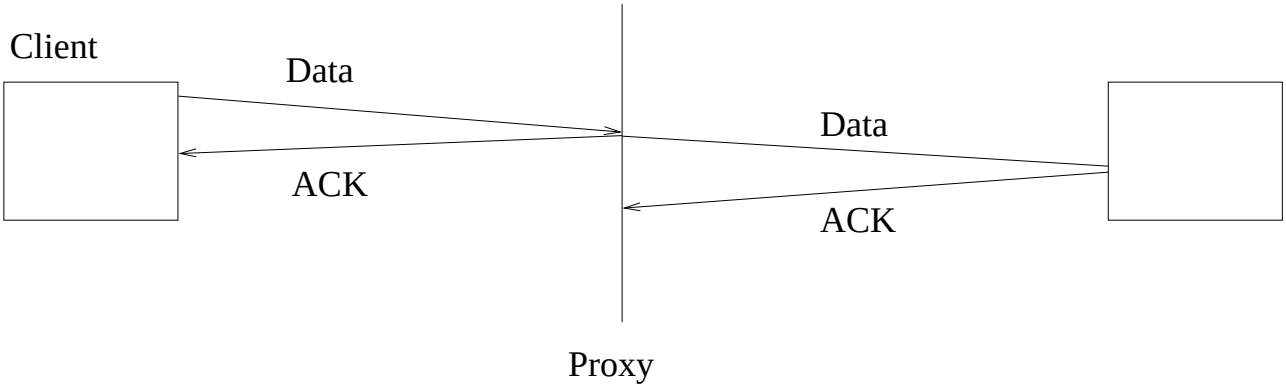
Per packet processing (IP)

- Reassembles IP fragments
- Removes IP options

Per connection processing (TCP)

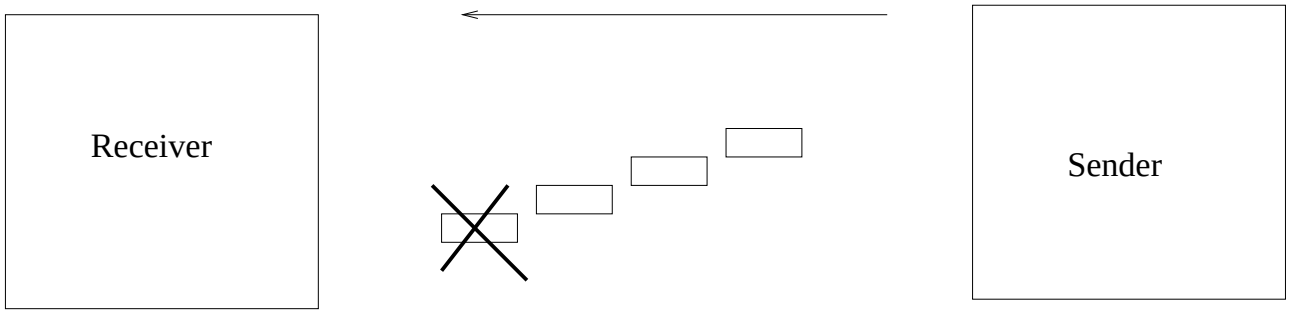
- Acknowledge data from client
- Ordering of TCP segments to client
- Assumes responsibility for TIME-WAIT connections

Acknowledge data from client



- Reduces the need for client buffering
- Proxy does retransmissions
- Breaks the end-to-end semantics
- Does not acknowledge SYN and FIN

Ordering of TCP segments

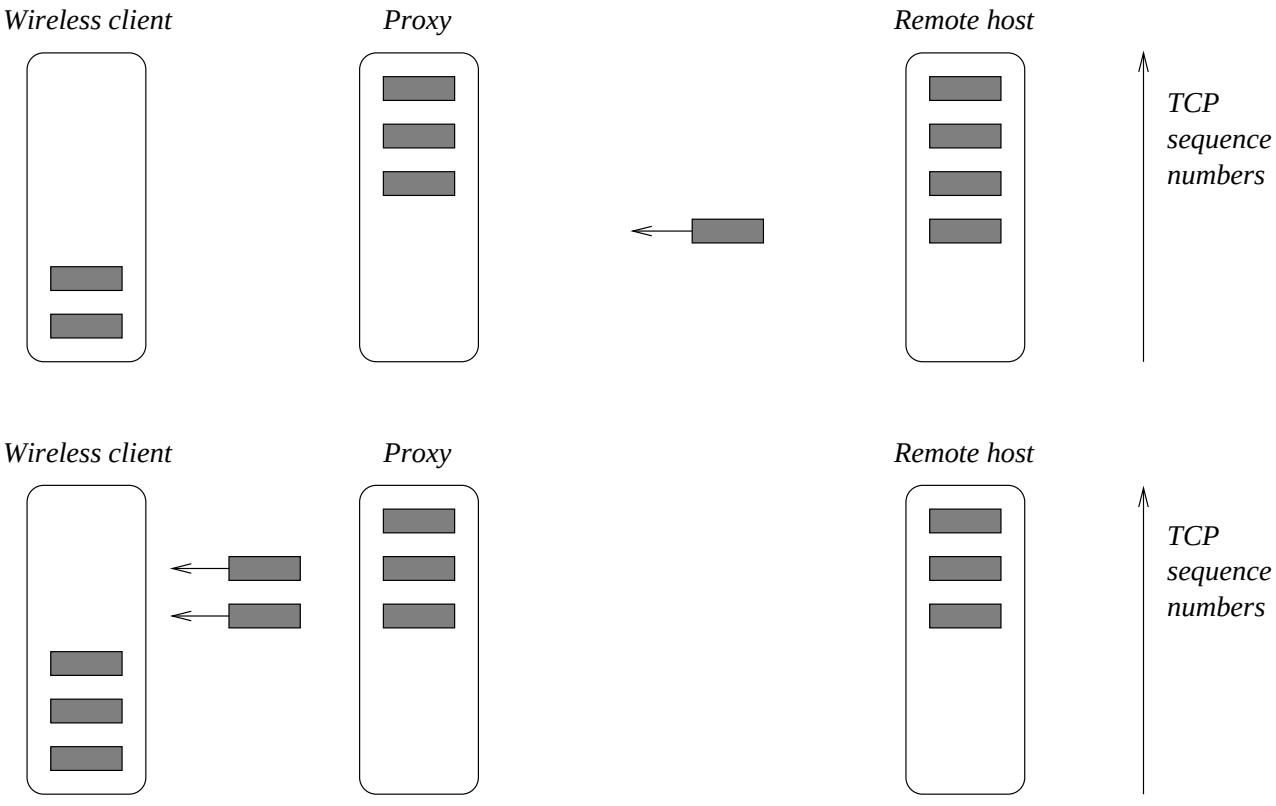


- Packet loss means out-of-sequence segments

- The client should buffer out-of-sequence segments

Ordering of TCP segments (contd)

- The proxy buffers out-of-sequence segments w/o forwarding
- The proxy does *not* acknowledge buffered segments
- When in-sequence segment arrives, buffered segments are sent to client



Assuming TIME-WAIT responsiblility

- After closing, a connection is in TIME-WAIT for $2 \times MSL =$ between 1 and 4 min
- Proxy follows TCP state transitions in client
- When client enters TIME-WAIT the proxy sends RST to client, killing TIME-WAIT connection
- Stops packets for $2 \times MSL$

lwIP — the small TCP/IP implementation

- Low RAM usage
- Specially designed API
- Small code size
- Portable, operating system emulation layer
- IP (+ forwarding), ICMP, UDP, TCP
- "Full" TCP implementation with RTT-estimation, congestion control, fast recovery/retransmit
- Can be run without proxy support, but runs better with proxy

RAM usage

- Buffer management designed for small RAM
- Small datastructures

API

- Does not require copying between application and stack
- Buffer management on application layer
- Possible to emulate BSD sockets

Size of compiled code

x86 (FreeBSD 4.1, gcc 2.95.2)

TCP	6584	48%
API	2556	18%
Support functions	2281	16%
IP	1173	8%
UDP	731	5%
ICMP	505	4%
Total	13830	100%

6502 (8-bit CPU)

TCP	11461	51%
Support functions	4149	18%
API	3847	17%
IP	1264	6%
UDP	1211	5%
ICMP	714	3%
Total	22646	100%

- FreeBSD TCP 27k, Linux TCP 39k

Future work

- Test and evaluate the proxy
- Incorporate wireless TCP throughput enhancements
- Header compression over shared media
- Test the TCP implementation in lwIP

Full report avaliable at `http://www.sics.se/~adam`