Shaping Process Semantics

Christoph Kirsch
Universität Salzburg

Joint work with Harald Röck
Monterey Workshop, Paris, October 2006

The Idea

- we apply *traffic shaping* technology known in the networking community to *software processes*
- software processes invoke *system calls* to access resources, perform I/O, etc.
- we see system calls as *network packets*

Process Shaping

- process shaping changes the *order* and *times* in which system calls (and potentially other *side effects* of processes) are handled before given to any performance-oriented kernel subsystems
- process shaping promotes more disciplined *system composition*

Application Processes

Timing Calls | I/O Calls | IPC Calls | Virtual Memory

Kernel

Process Shaping

Interrupt Handling | I/O Scheduling | IPC Handling | Memory Management

Hardware

Semantics

Performance

Proposal

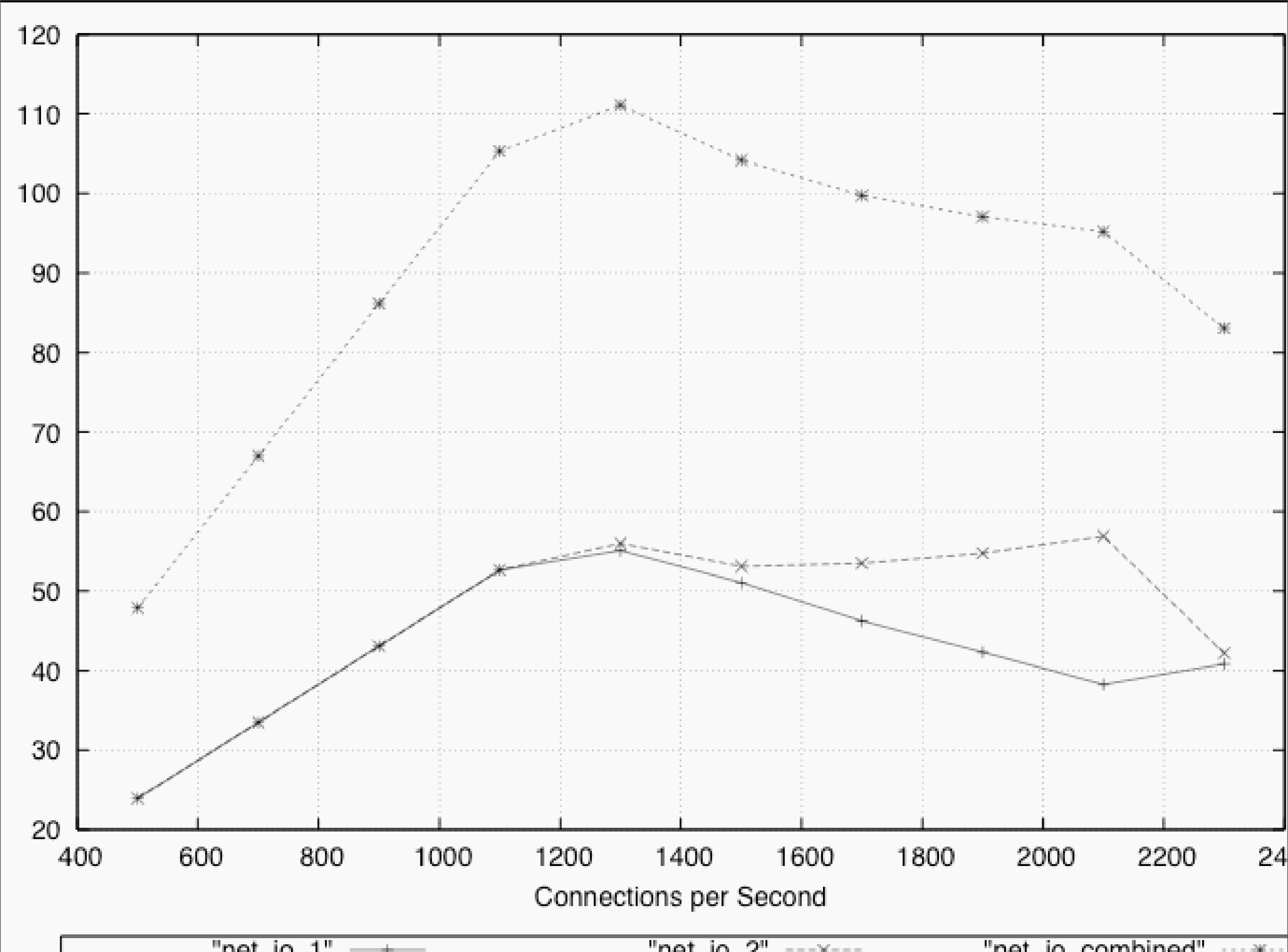
- we propose the notion of *process shaping* to complement, not replace, the notion of *serving* processes *as fast as possible*
- we advocate a shift in research attention from performance- to *semantics-oriented* handling of software processes

Claim

- we claim that *faster* processors, more efficient scheduling, and lower kernel *latency*, in analogy to *shorter* packet transmission times, will make process shaping increasingly effective
  - see ATM versus Gigabit Ethernet
- note that used-to-be-exotic real-time patches increasingly make their way into general-purpose operating systems

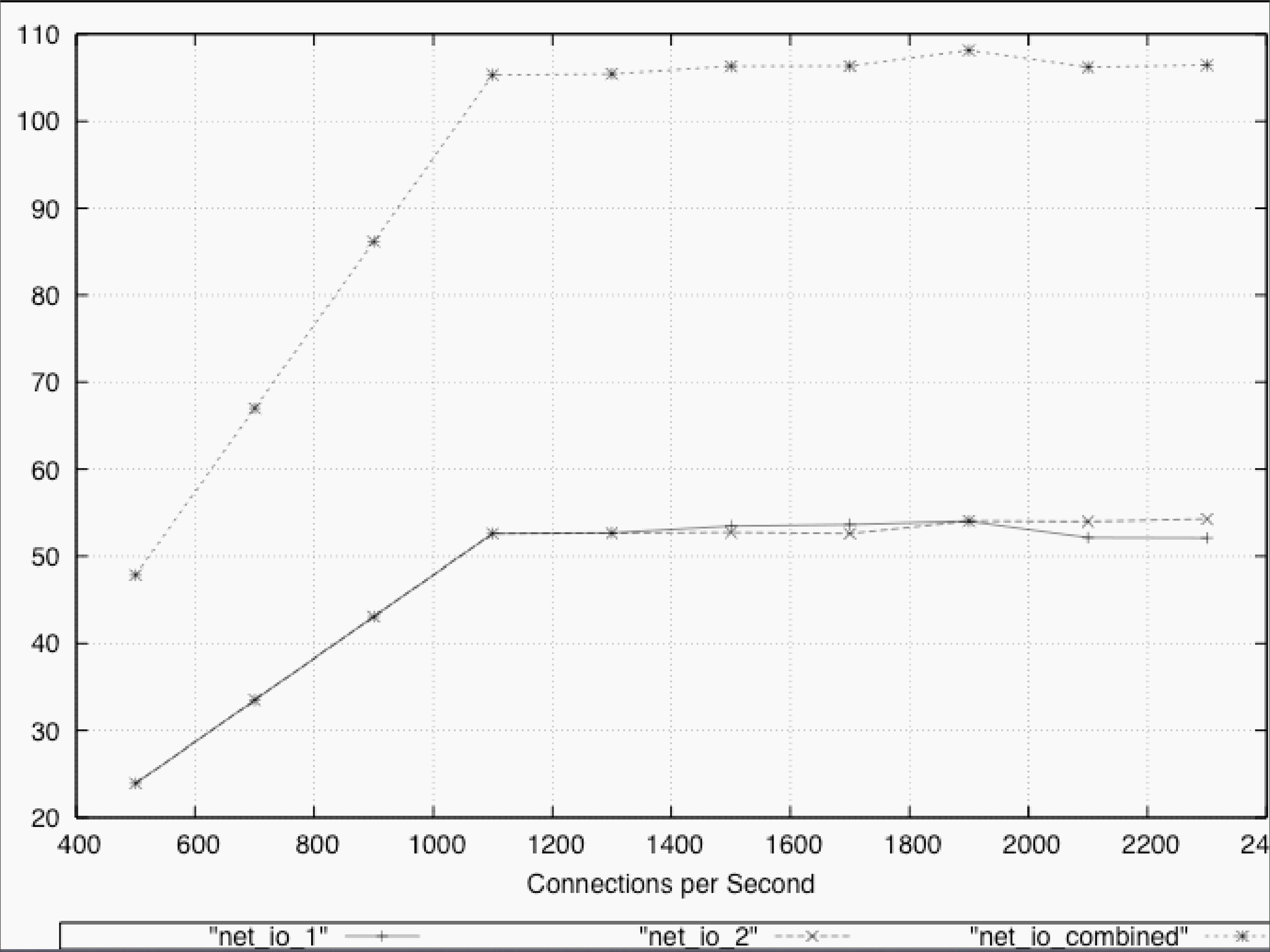
Experiment I

- we run two separate web server processes on an *unmodified* Linux 2.6 server machine with Gigabit Ethernet
- two client machines generate workload by requesting the same and thus *cached* 380KB file


120
110
100
90
80
70
60
50
40
30
20
400
600
800
1000
1200
1400
1600
1800
2000
2200
24
Connections per Second
"net_io_1"
"net_io_2"
"net_io_combined"

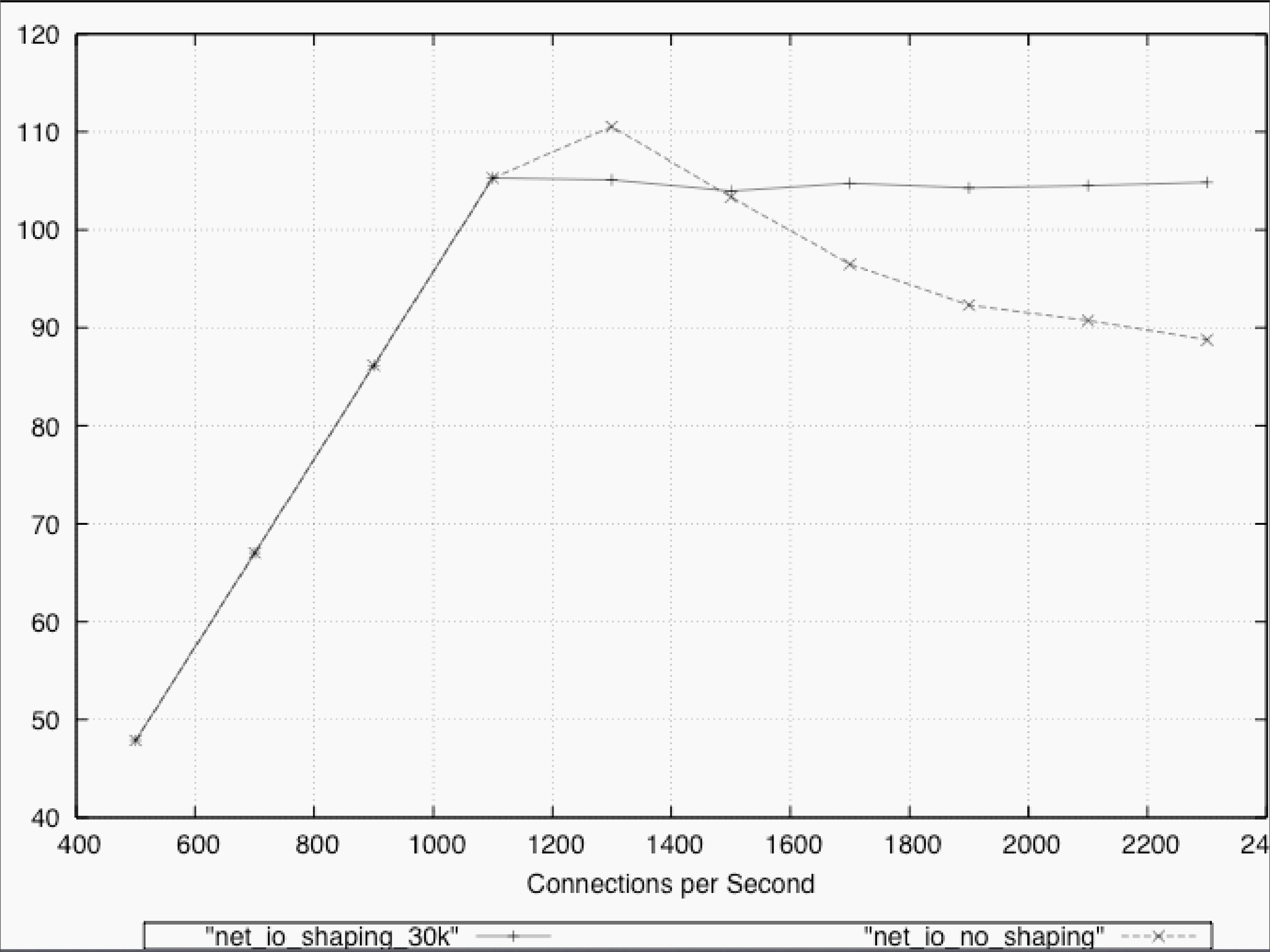
Experiment II

- we run two separate web server processes on a *process-shaping* Linux 2.6 server machine with Gigabit Ethernet
- two client machines generate workload by requesting the same and thus *cached* 380KB file


110
100
90
80
70
60
50
40
30
20
400
600
800
1000
1200
1400
1600
1800
2000
2200
24
Connections per Second
"net_io_1"
"net_io_2"
"net_io_combined"

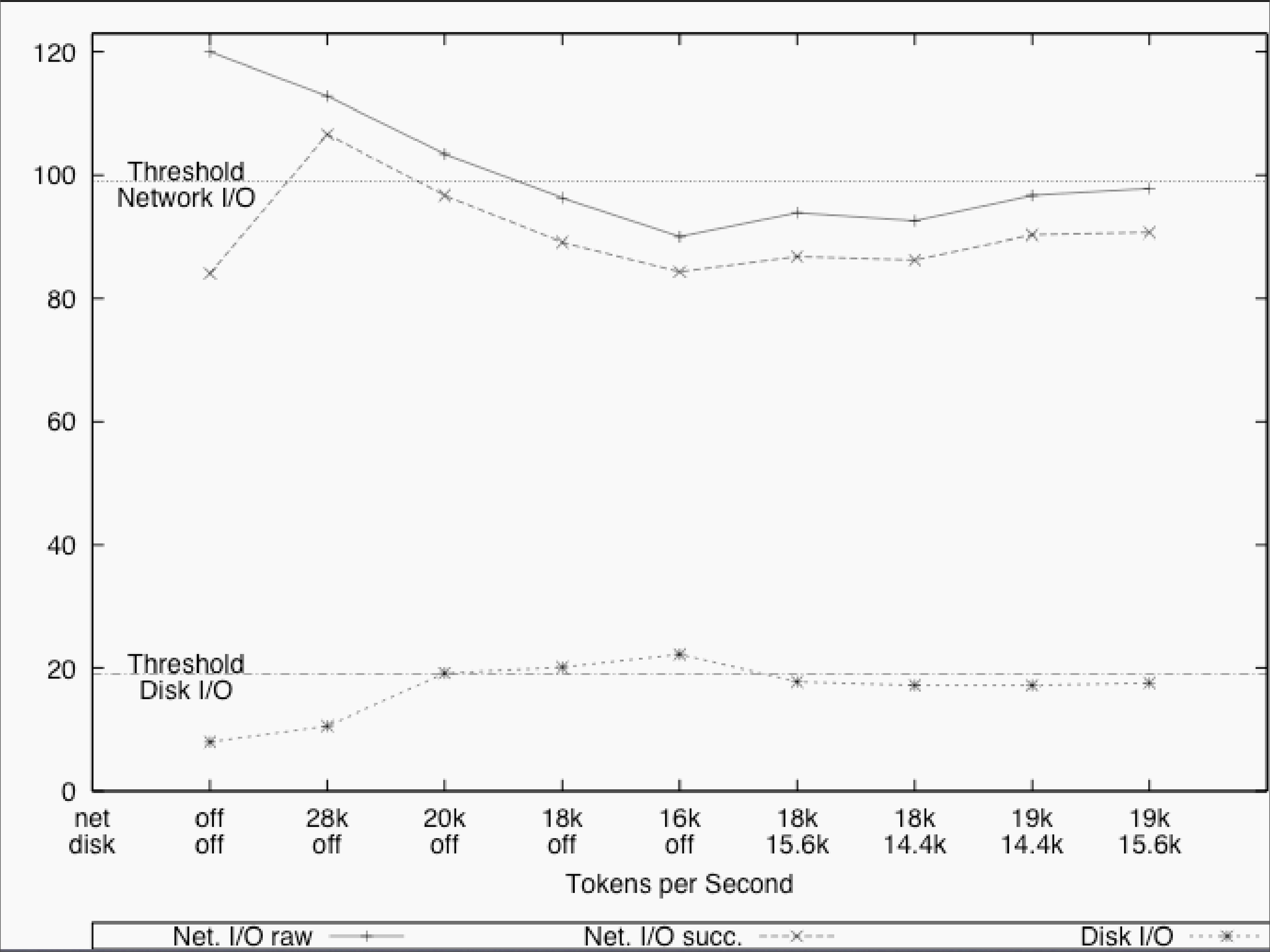
Experiment I+II

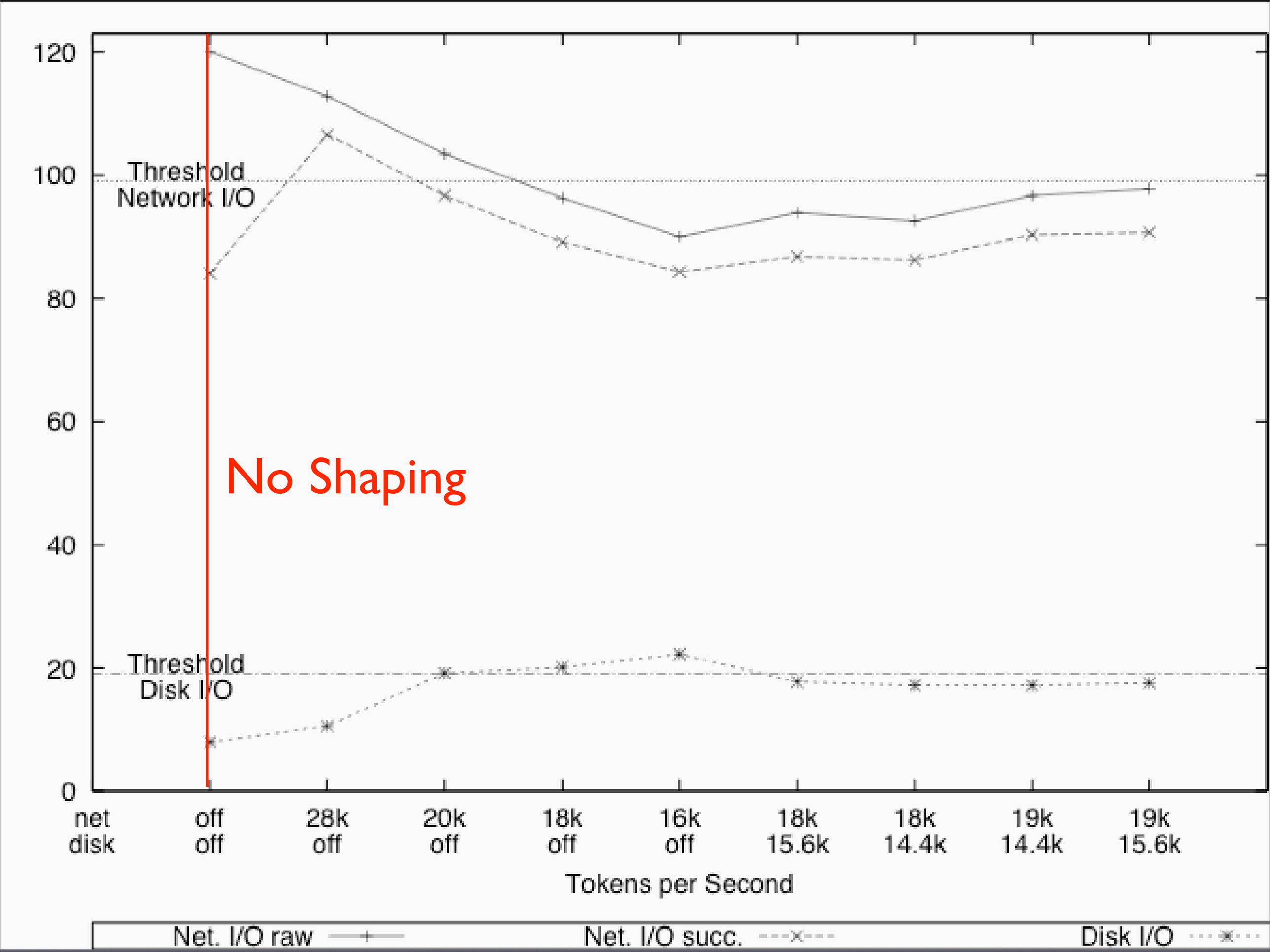
- higher total peak performance without process shaping
- but total peak performance more robust with process shaping

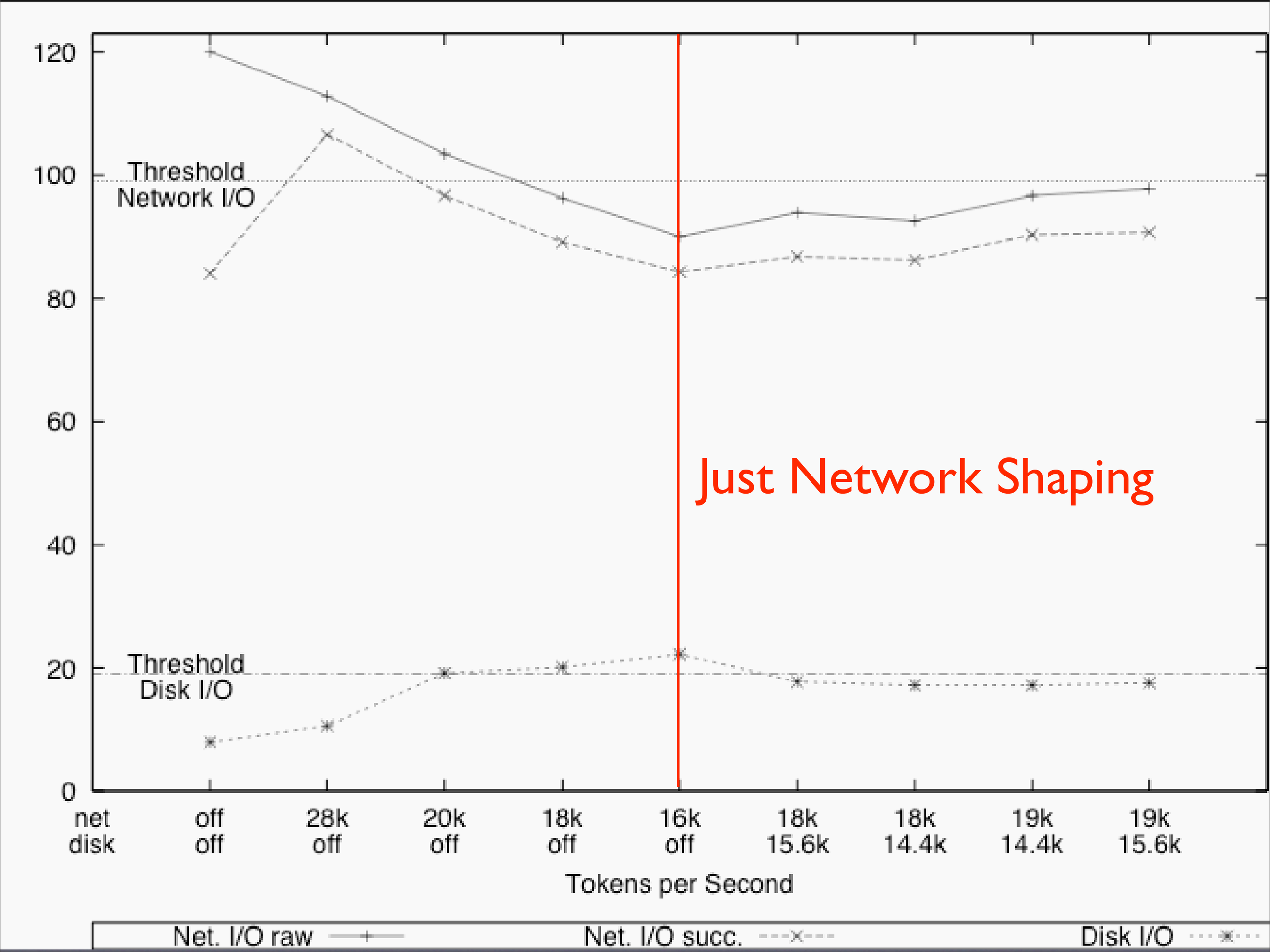

120
110
100
90
80
70
60
50
40
400
600
800
1000
1200
1400
1600
1800
2000
2200
24
Connections per Second
"net_io_shaping_30k"
"net_io_no_shaping"

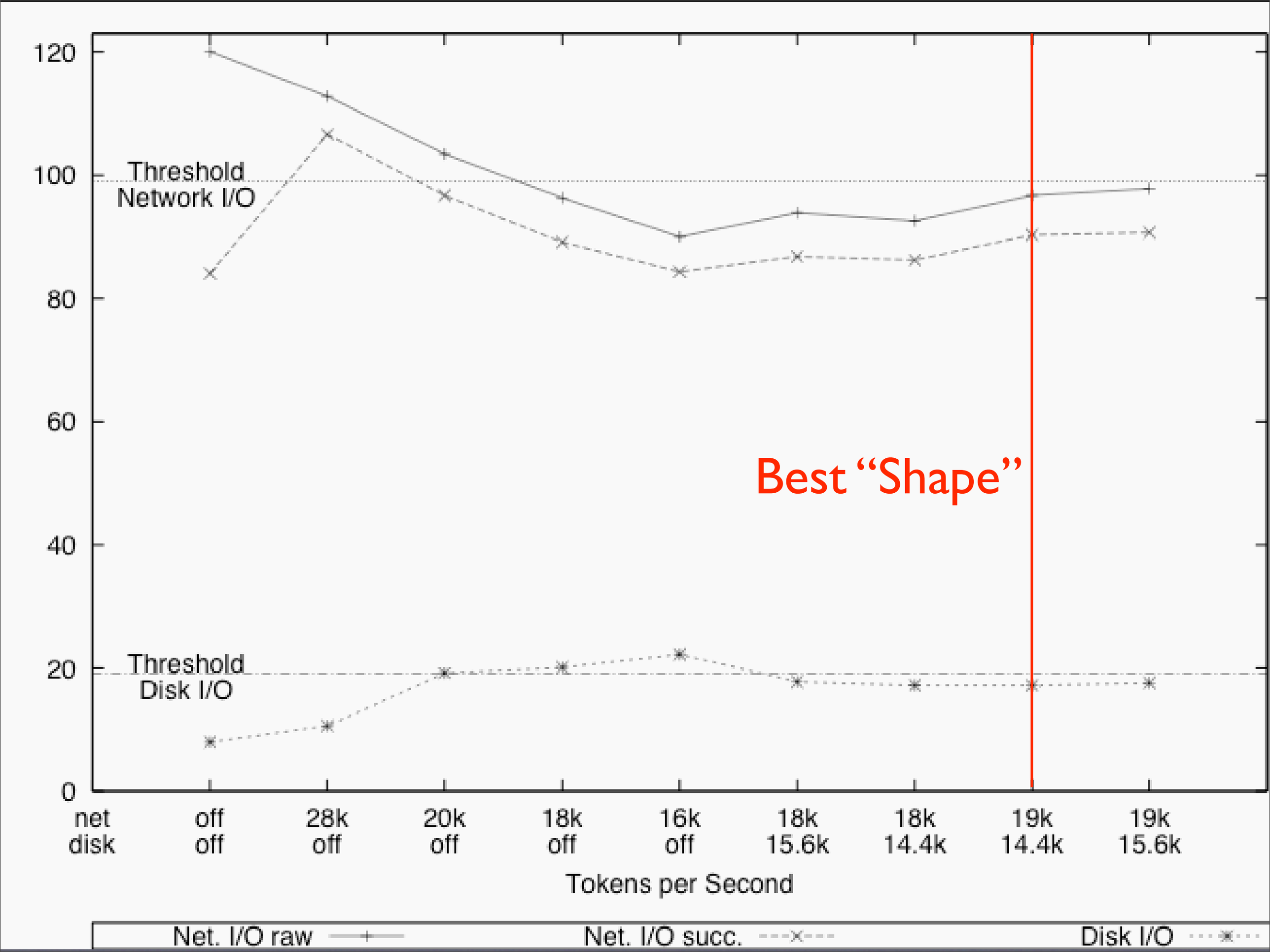
Experiment III

- we run a *video-streaming* server on a *process-shaping* Linux 2.6 server machine with Gigabit Ethernet
- to generate background *network traffic*, we also run one of the web servers of experiment II on the same machine
- to generate background *disk traffic*, we also run several web servers processing requests for a non-cached 1GB file


120
100
80
60
40
20
0
Threshold
Network I/O
Threshold
Disk I/O
net
disk
off
off
28k
off
20k
off
18k
off
16k
off
18k
15.6k
18k
14.4k
19k
14.4k
19k
15.6k
Tokens per Second
Net. I/O raw
Net. I/O succ.
Disk I/O


120
100
80
60
40
20
0
Threshold
Network I/O
Threshold
Disk I/O
No Shaping
net
disk
off
off
28k
off
20k
off
18k
off
16k
off
18k
15.6k
18k
14.4k
19k
14.4k
19k
15.6k
Tokens per Second
Net. I/O raw
Net. I/O succ.
Disk I/O


120
100
80
60
40
20
0
Threshold
Network I/O
Threshold
Disk I/O
Just Network Shaping
net
disk
off
off
28k
off
20k
off
18k
off
16k
off
18k
15.6k
18k
14.4k
19k
14.4k
19k
15.6k
Tokens per Second
Net. I/O raw
Net. I/O succ.
Disk I/O


120
100
80
60
40
20
0
Threshold
Network I/O
Threshold
Disk I/O
Best "Shape"
net
disk
off
off
28k
off
20k
off
18k
off
16k
off
18k
15.6k
18k
14.4k
19k
14.4k
19k
15.6k
Tokens per Second
Net. I/O raw
Net. I/O succ.
Disk I/O

Future

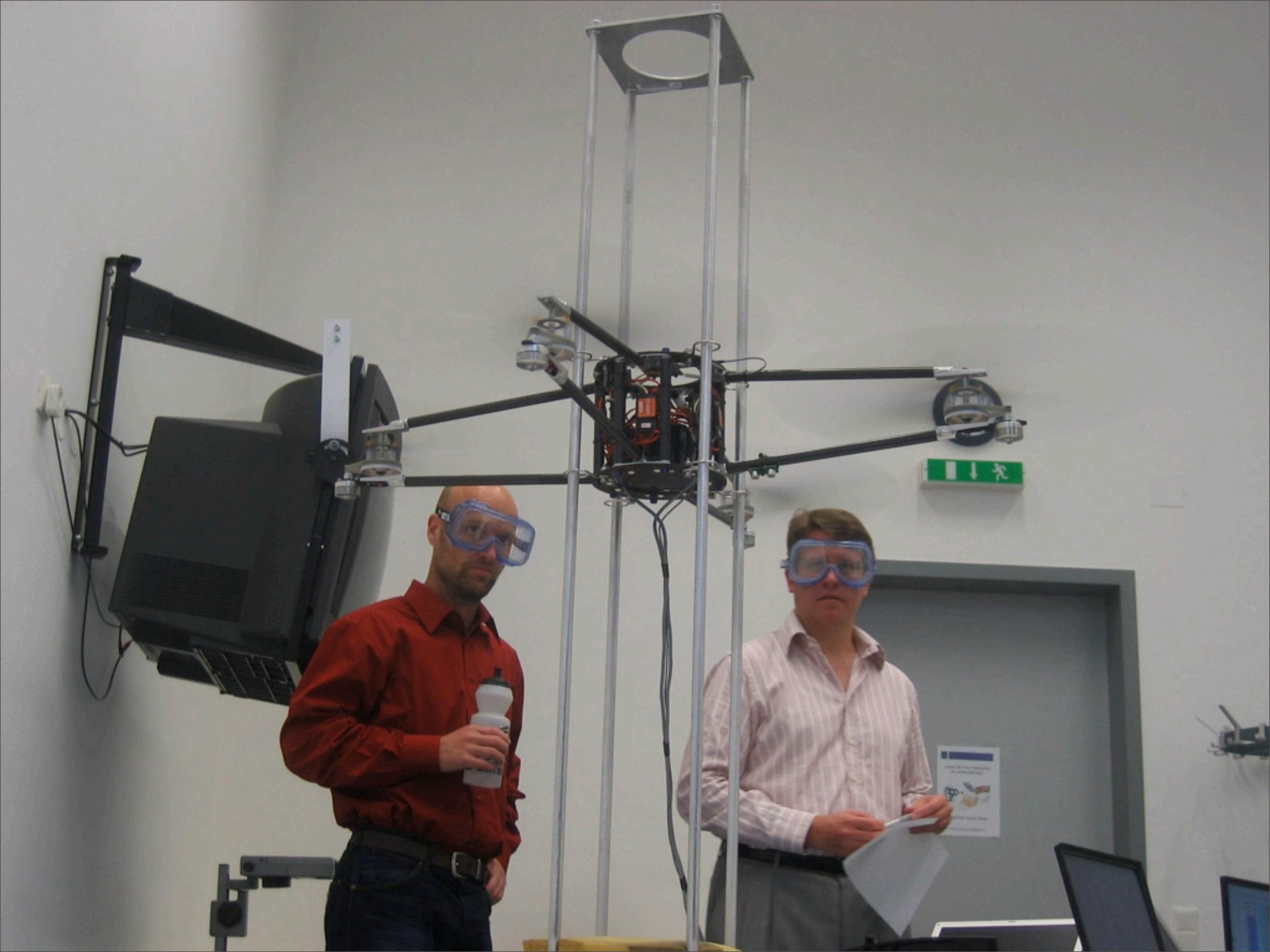
- Multimedia (*Soft* Real Time): Can kernel-level process shaping automatically find the best "shape"?
- Control (*Hard* Real Time): Can kernel-level process shaping provide sufficient real-time guarantees?

Experiment IV

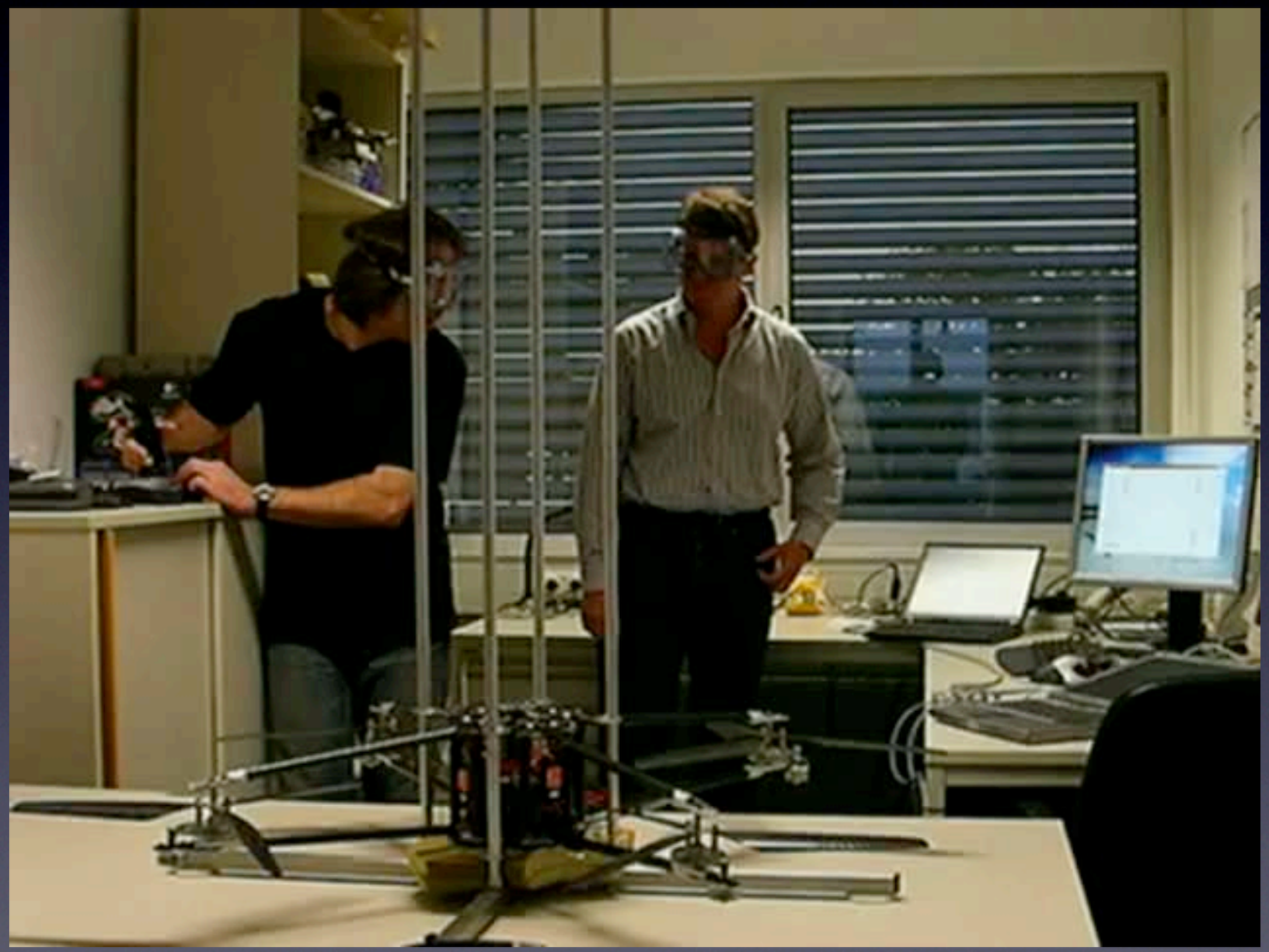
- we run helicopter flight control software written in *Java* on IBM's commercial J9 JVM with the real-time garbage collector *Metronome* on top of a Linux 2.6 machine with real-time patches applied to the kernel
- joint work with J. Auerbach, D. Bacon, H. Röck, and R. Trummer

The JAviator Project

javiator.cs.uni-salzburg.at



First All Java Flight



Thank you