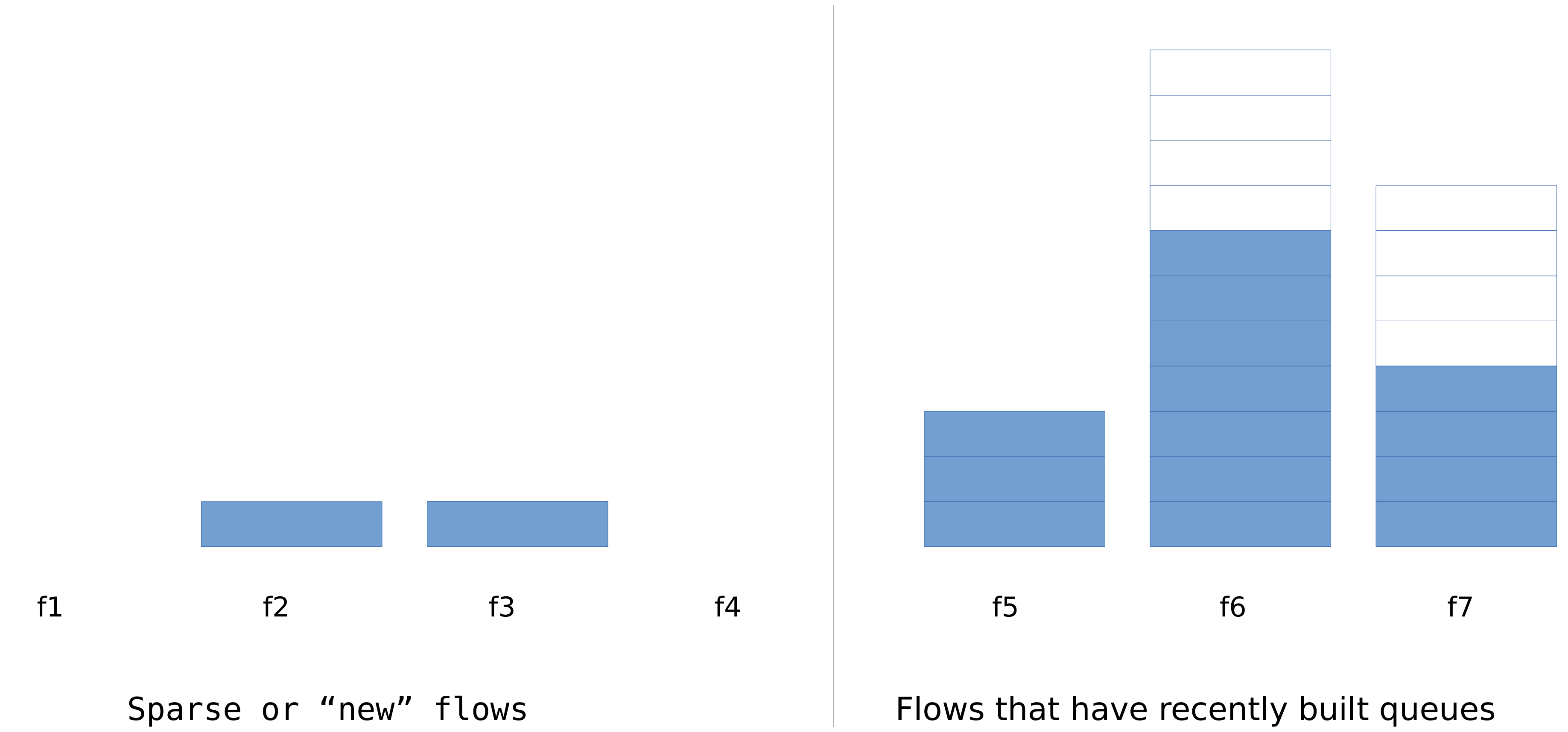
fq_codel status

Jim Gettys
Bell Labs

fq_codel Packet Scheduling: similar to airport security queues...

Flows are identified. The first packet(s) from new flows are scheduled before packets from flows that have built a queue. If a flow's queue empties, it is eligible to again be considered a new flow. CoDel is applied to control the queue length in any flow. Result: timely delivery, and mixing of flows



fq_codel works and solves it's intended problem

- fq_codel even works decently well with BitTorrent traffic, without diffserv marking (but we could do better if traffic were marked)
- Even *without any packet classification* real time applications such as VOIP and gaming “work” under load in real world environments
- We argue that most needs for classification are not necessary once running fq_codel; but hints are always useful. At the very edge of the network, marking and classification may have enough value to bother to use, primarily for priority access to the medium and providing strong usage guarantees

Linux Implementation

- In Linux since Linux 3.3, since July 2012
- Widely tested under many environments, since you can often enable it without building your own kernel
- Now “on by default” in OpenWrt on all interfaces
- In general, we're happy, and performance is great
- So far, various minor tweaks to fq_codel have not tested out enough to be worth worrying about
- But as always, there are issues; real implementation != algorithm...

Next steps for Linux

- What should the “default” queue discipline(s) and configuration for Linux be, to replace PFIFO_FAST?
  - fq_codel is “first, do no harm”, but may not be effective in some environments (e.g. data centers)
- This discussion is underway, but will take meetings at the Linux Plumber's Conference to socialize
- Device drivers...

Device Drivers

- Kernel interface to drivers should be rethought
- Input buffering can also be an issue; people often overlook it
  - fq_codel can only control buffers under its control: device drivers have additional buffering: e.g. transmit/receive rings, error correction buffering
- Offload engines cause “interesting” *as in the Chinese curse) problems, on many paths, particularly in concert with certain broadband gear that merges TCP acks: fq_codel helps this situation, but offload engines get in the way

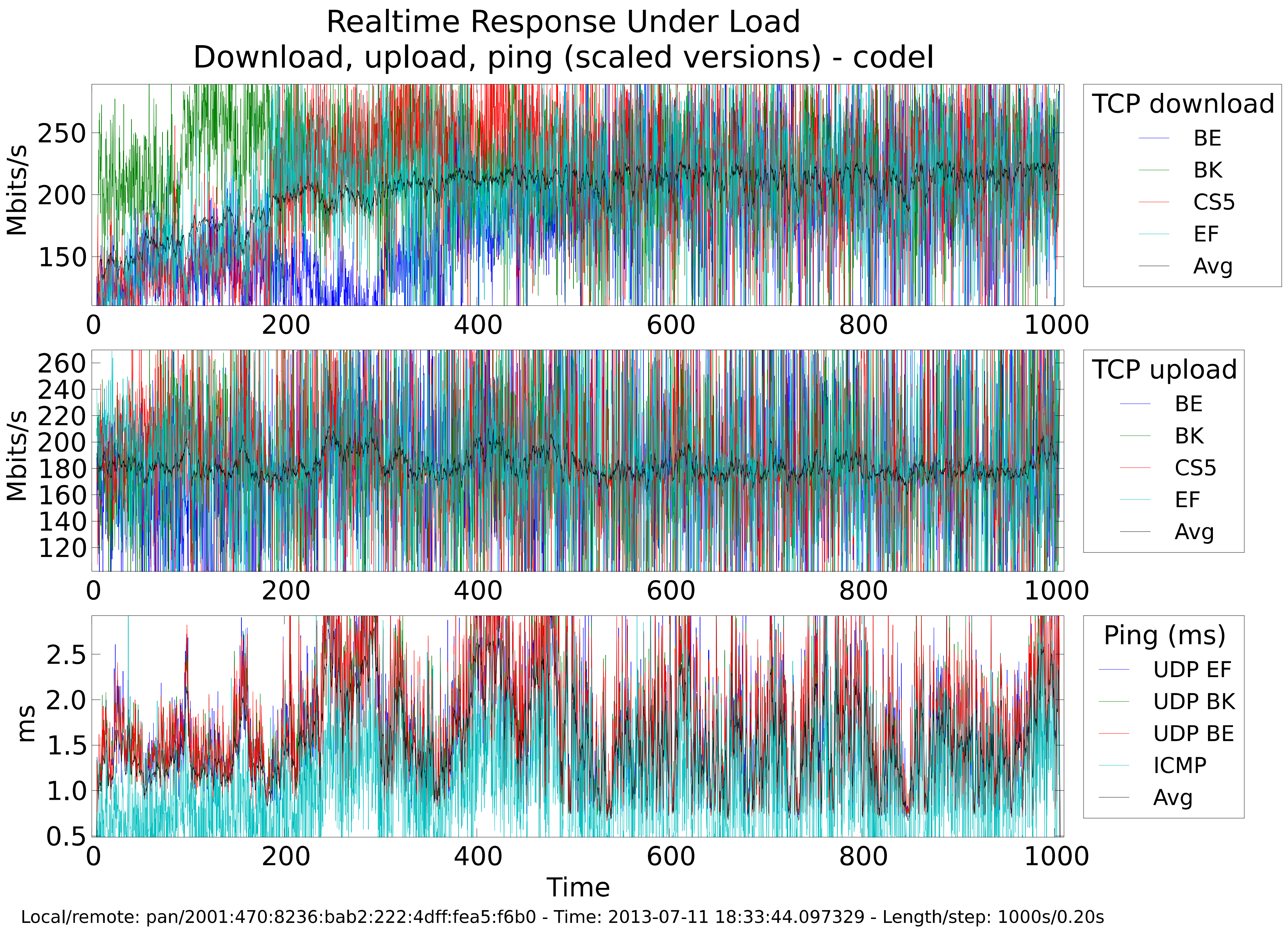
Linux Device Drivers

- Ethernet has the BQL framework for Ethernet transmit ring control
- But we don't have an equivalent for other network frameworks: e.g. VDSL
- 802.11 is a particular challenge due to aggregation and also input buffering for error correction
- Similar problems probably exist in LTE drivers in handsets (and base stations)

Input Buffering

- Often overlooked, but it can become a serious issue: e.g. 802.11 drivers, or....
- If your processor is wimpy, the bottleneck can shift to your input buffer from your output buffer
  - fq_codel needs “ingress” timestamping; the current Linux fq_codel implementation timestamps on entry to fq_codel's queue, and this can be a problem if your hardware receives packets for a “long time” without processor intervention
  - Timestamps should be applied in a “timely” way on ingress to a system, rather than when added to the output queue

Driver bug? Input buffering?



Future work

- Wireless
  - is inherently highly variable bandwidth for nodes, on low time scales
  - How does CoDel's strategy work on such networks? Do we need something better?
- Scaling
  - In great shape @ 10G: it's 2% of a single modern CPU core, while increasing network utilization
  - Mixing flows is good for the health of the network
  - Can you apply the algorithm all the way up to 1000G routers? We think so, but careful cache analysis needs to be done.