Experiences with Interactive Video Using TFRC

Alvaro Saurin, Colin Perkins
University of Glasgow, Department of Computing Science

Ladan Gharai
University of Southern California, Information Sciences Institute
Talk Outline

• Aims and objectives
• Implementation and performance of TFRC
• Implications for real-time video
  – Protocol issues
  – System design issues
  – Experimental results
• Open issues and implications for DCCP
Aims and Objectives

• Evaluate performance of interactive video conferencing systems running over congestion controlled transport
  – Implemented video conferencing tool
    • PAL/NTSC format video
    • Motion-JPEG compression ⇒ responsive, low compression delay
    • Typical data rate ~10s Mbps
  – User space implementation of TFRC, sending feedback within RTCP, data in modified RTP packets
    • draft-ietf-avt-tfrc-profile-05.txt
    • DCCP implementations not available when work started
    • Expect many results applicable to DCCP implementation, although a kernel implementation might have better timing characteristics
  – Experiments
    • Over Internet: Arlington, VA ↔ Glasgow ↔ Helsinki
    • Using local test bed (FreeBSD dummynet)
Implementation

- TFRC implementation can be done at application level, part of existing RTP stack
- Four basic functions in feedback loop:

```
  data
   ↓
Rate Calculation
  ↓
Packet Spacing
  ↓
  Sender

  ↓
Feedback
  ↓
Loss Detection
  ↓
Receiver
```

- Challenges:
  - Accurate packet spacing at sender
  - Timely feedback
Implementation: TFRC sender

- High performance video requires small inter-packet interval
- Difficult to accurately schedule packets
  - Due to inaccurate wakeup after sleep, thread scheduling issues

Example: 3.5ms RTT @ 8Mbps

Errors in inter-packet spacing on same order of magnitude as the RTT
Implementation: TFRC receiver

• Similar issues with slow wakeup
  – System slow to schedule thread on expiry of feedback timer
  – 10ms wakeup latency not uncommon
  – Significantly delays feedback

• Timing inaccuracy in sender and receiver poses a significant challenge to stable TFRC implementation
Experimental Performance: TFRC

3.5ms – 8000 Kbit/s

20ms – 3000 Kbit/s

100 ms – 500 Kbit/s

200 ms – 200 Kbit/s

TCP Flows = 1

TCP Flows = 2

TCP Flows = 4

TCP Flows = 8
Experimental Performance: TFRC

• Observe poor stability with short RTT:

• Issues:
  – Bursty sending behaviour
    • Packets sent in bursts spaced around wakeup intervals
    • Degenerates into something similar to a window-based approach
    • May be simpler just to use a window based protocol?
  – Slow feedback
    • With 10ms wakeup latency and 3.5ms RTT, possible for feedback to be delayed >2RTT due to inaccuracies
    • Will force sender to halve sending rate

• Have found stability difficult to achieve with RTT < 10-20ms
From Glasgow, the RTT to much of the UK is within problematic region

- Straight forward to add smoothing to protocol
  - Reduces responsiveness and fairness to TCP
  - Kernel implementation of TFRC likely more accurate timing ⇒ smoother
Implementation: Video Transmission

- Capture and transmission operate on different time scales
  - Slow bursts of arrivals from codec
  - Fast, smoothly paced, transmission

- Mismatched adaptation rates
  - TFRC ⇒ O(round-trip time)
  - Codec ⇒ O(inter-frame time)
  - Relies on buffering to align rates, varies codec rate

- Capture and encoding process causes timing problems:
  - Capture DMA operation can disrupt other bus accesses
  - Encoding uses significant amounts of processor time
    - M-JPEG currently, other codecs likely much much worse
    - Linux general purpose scheduler barely adequate to get predictable thread scheduling in this environment; real-time scheduler difficult to tune/debug

- Sender dynamics difficult to tune and debug
Experimental Performance: Video

100ms RTT, 800kbps bottleneck, 10 fps M-JPEG
Testing in dummynet

Desired vs. actual sending rate

Best case: RTT and frame rate match
Experimental Performance: Video

- Poor man’s video quality metric:
  - Peak Signal to Noise Ratio (PSNR)
  - Significant variation in quality over session lifetime
    - Changes in input source requires a variable output rate
    - Constrained to be smooth by TFRC ⇒ quality varies instead
- Also see packet losses due to rate limit at sending buffer
  - Could be solved by faster codec adaptation
  - But: requires codec that can change compression ratio within a frame
    - Effect on quality unclear; implementation challenge
Issues: Slow Start

- Slow start requires an application to send at a low initial rate, increasing exponentially each round-trip time where no loss is reported
  - Duration of slow start period depends on network conditions; unpredictable

- Video codec must be capable of such a rapid increase in sending rate whilst maintaining reasonable picture quality
  - Requires a highly scalable codec, capable of varying compression ratio on the order of network RTT
    - i.e. while coding a frame, since RTT likely doesn’t match frame rate
    - Not clear this is feasible
  - Current implementation generates dummy data instead
    - Seems wasteful, but can cover call setup delay
Issues: Steady State

• Application required to send at a roughly constant rate, based on average loss rate observed
  – Transmission rate narrowly bounded
    • Large bursts above the prescribed rate must be avoided due to insufficient capacity; less aggressive senders will be “beaten down” by TCP traffic as consequence of the TFRC algorithm
    • Imposes constraints on when a codec can change its rate
    • Given sufficient buffering, and use of dummy data, is possible to meet rate constraints; not clear feasible for interactive systems
  – Difficult to accurately match transmission rate
    • Requires codec that can change rate on O(RTT) timescale
      – High frame rate; or codec that can vary compression within a frame
    • Requires accurate feedback timing
    • Problems with short RTT
Conclusions

• Initial experiments raise more questions than they answer
  – Likely possible to run video over TFRC, with more sophisticated codecs
    • Impact on perceptual quality of implied quality variation unclear
    • Likely easier as video quality, frame-rate and network bandwidth increase
  – Slow start very problematic
    • Codecs don’t adapt in an appropriate way
  – Given difficulty in matching rate, and resulting bursty behaviour, not clear that window based congestion control wouldn’t be more appropriate
    • To what extent is sending dummy data appropriate?

• DCCP a good base for experimentation
  – Not clear we understand problem sufficiently to give production quality advice on implementation of congestion controlled interactive video on TFRC
  – Changes in 3448bis may help; need more study