

Best-Effort Multimedia Networking

Dealing with Delay-Jitter

Kevin Jeffay

Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
November 11, 1999

<http://www.cs.unc.edu/~jeffay/courses/comp249f99>

1

Best-Effort Multimedia Networking

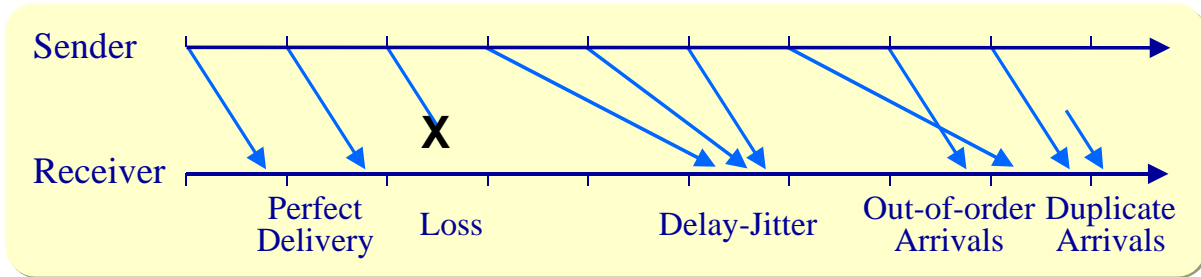
Outline

- ◆ IP message delivery semantics
 - » The four common Internet pathologies
- ◆ Ameliorating the effects of delay-jitter
 - » “60 ways to queue & play your media samples”
- ◆ Ameliorating the effects of packet loss
 - » Recovery of lost samples through retransmission
 - » Recovery of lost samples through the addition of redundant information
- ◆ Congestion control
 - » Adaptive media scaling and packaging

2

Best-Effort Multimedia Networking

Internet pathologies



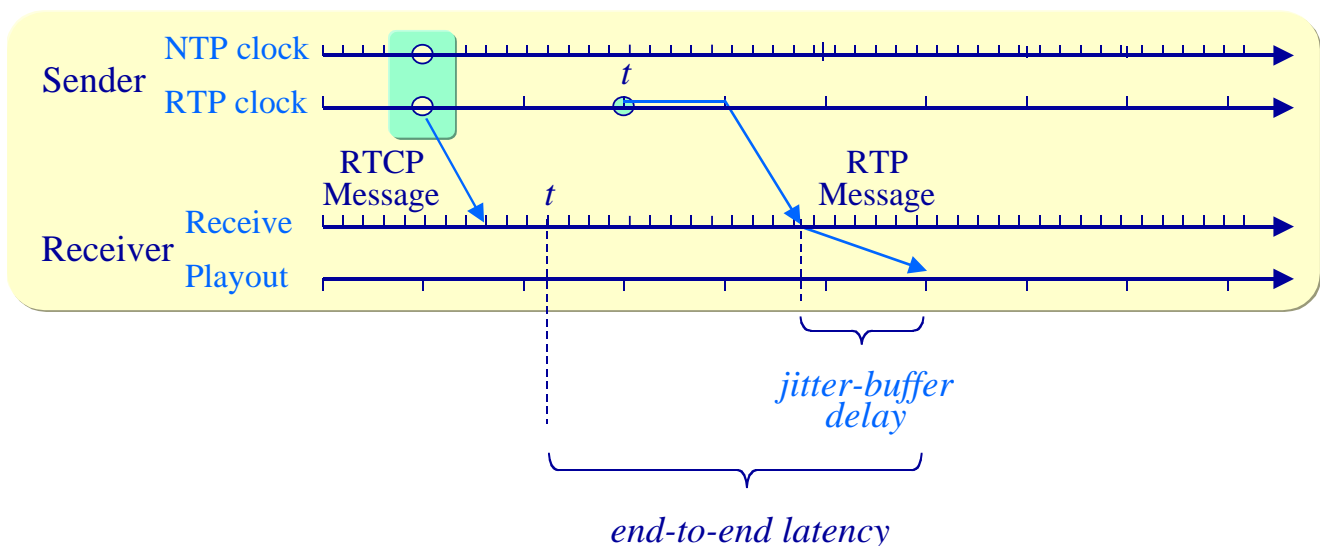
- ◆ Delay-jitter
 - » Managing a trade-off between end-to-end latency and continuous playout
- ◆ Loss
 - » Proactively control through forward error correction
 - » Reactively control through retransmission
- ◆ Out-of-order arrivals
 - » Assume out-of-order sample is lost
 - » Assume sample is late
- ◆ Duplicate arrivals
 - » Do we care?

3

Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ When the first media sample arrives, should it be played or enqueued?

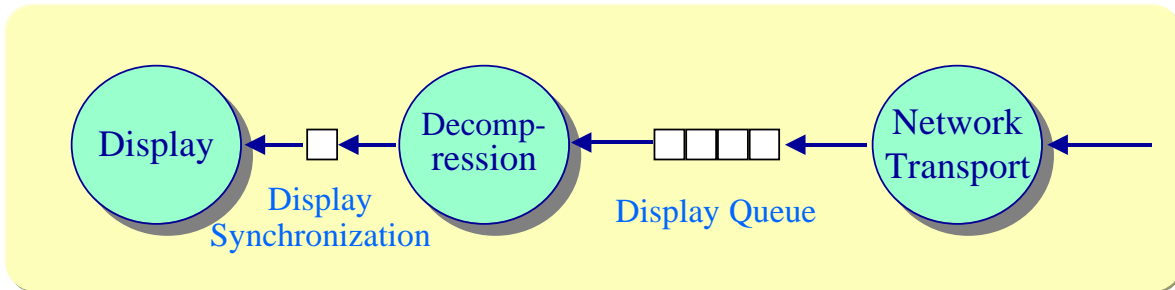


4

Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ When the first media sample arrives, should it be played or enqueued?



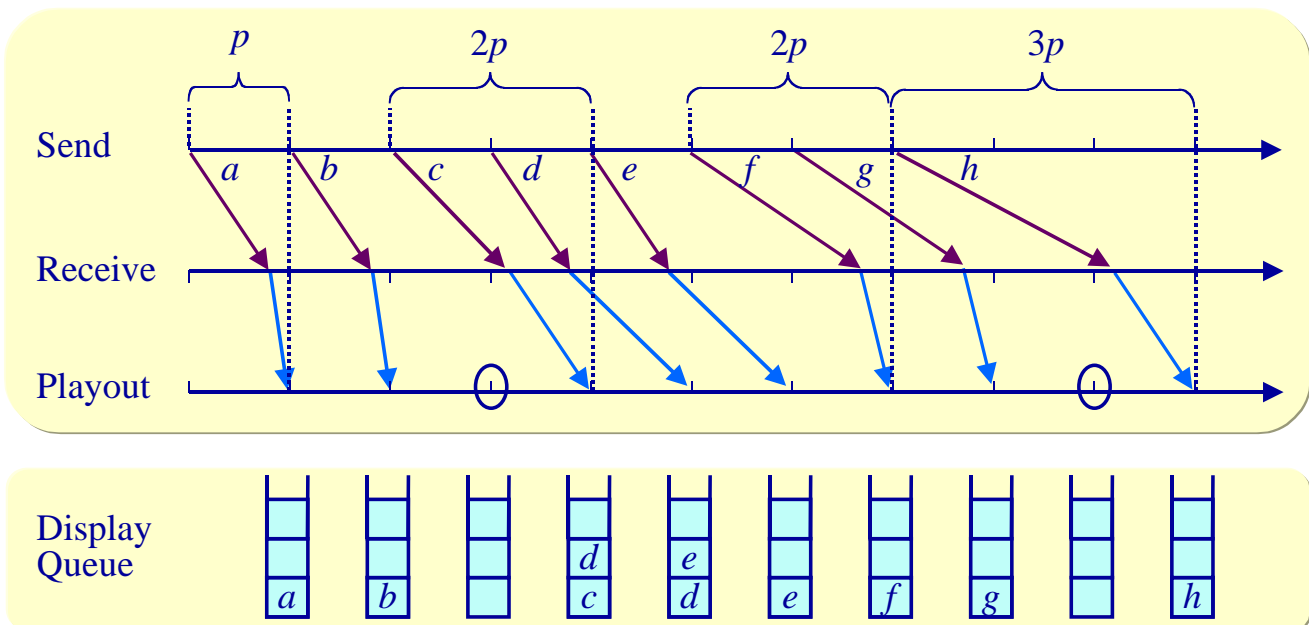
Receiver's processing pipeline

5

Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ When the first media sample arrives, should it be played or enqueued?
 - » Playing the sample ensures minimal end-to-end latency...

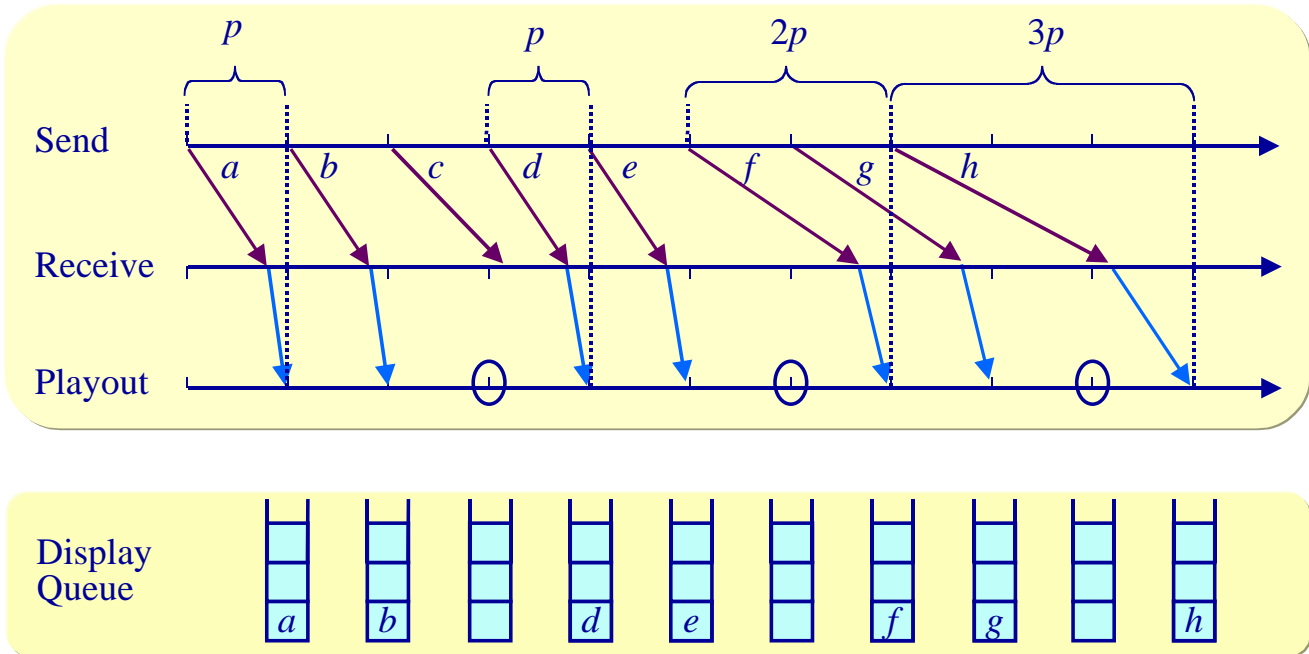


6

Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ Samples that arrive “too late” may be discarded

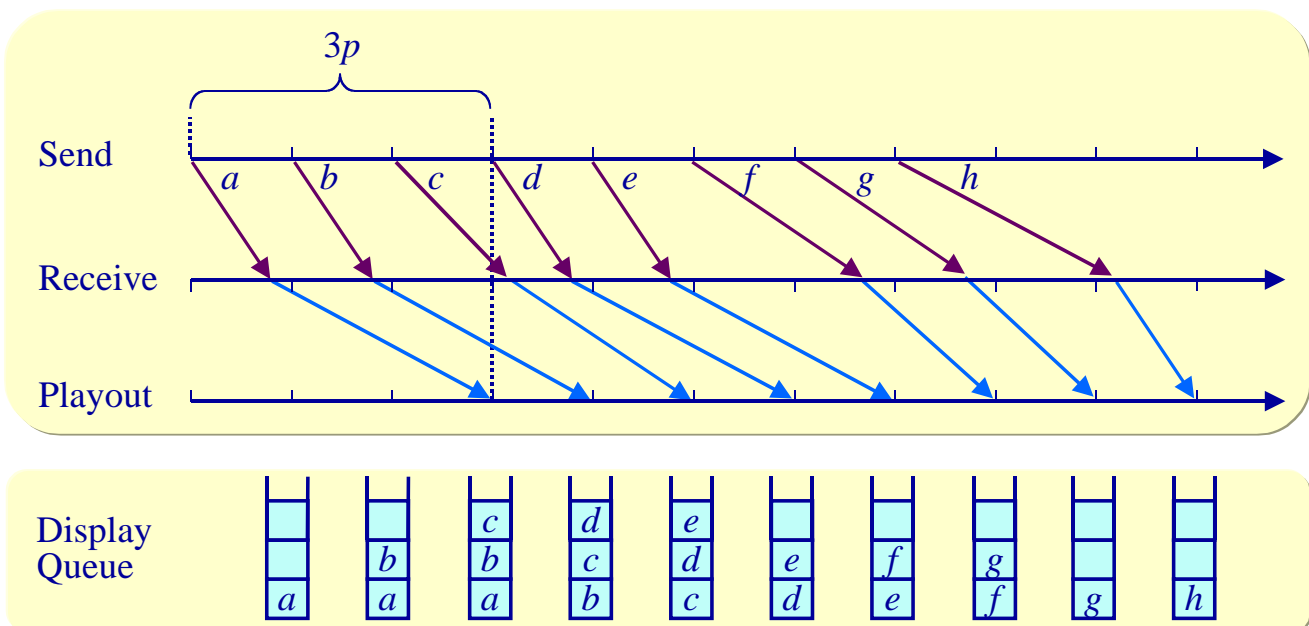


7

Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ Enqueueing the first sample better ensures continuous playout...

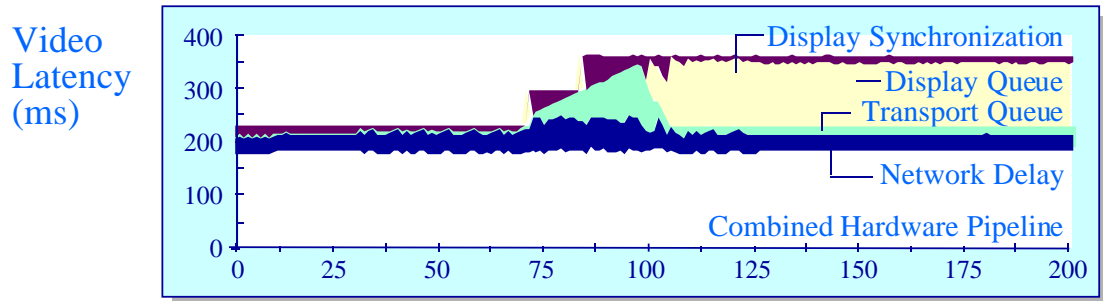


8

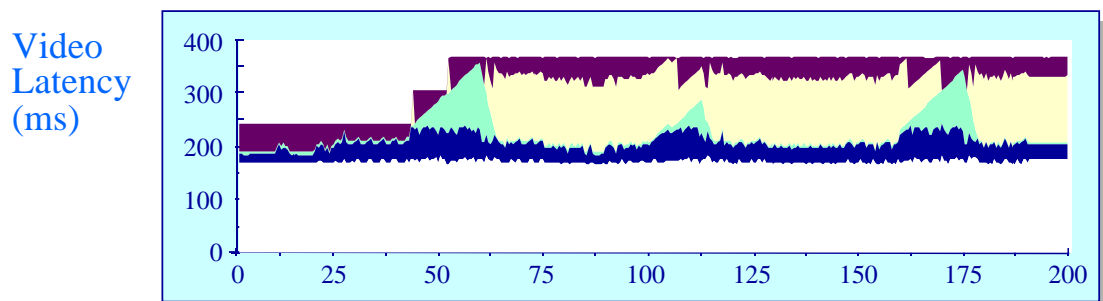
Ameliorating the Effects of Delay-Jitter

Trading-off end-to-end latency for continuous playout

- ◆ A large display queue is “bad”



- ◆ A large display queue is “good”!

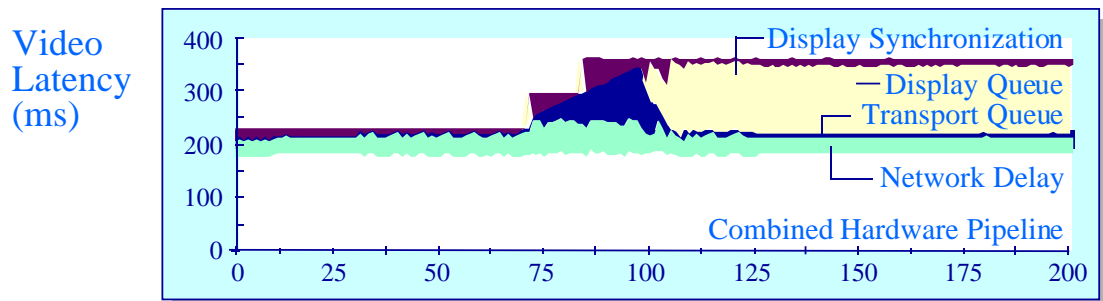


9

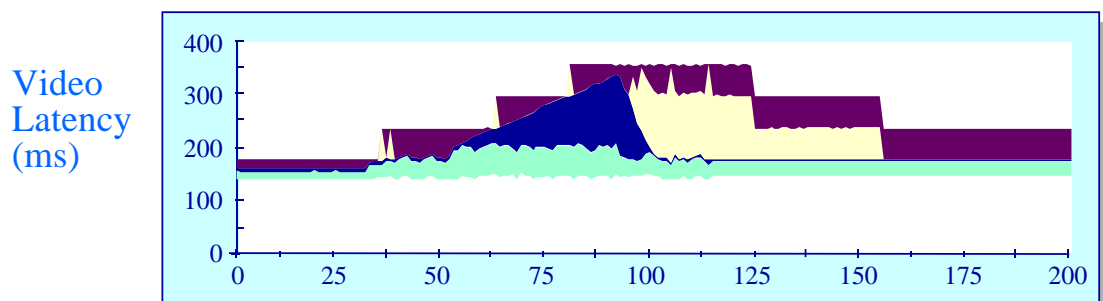
Principles of Delay-Jitter Buffering

Sample discarding

- ◆ A large display queue is “bad”



- ◆ Purposefully throwing away samples reduces latency

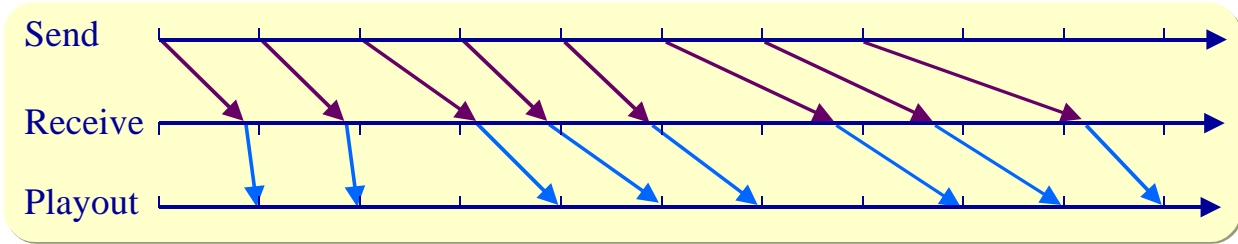


10

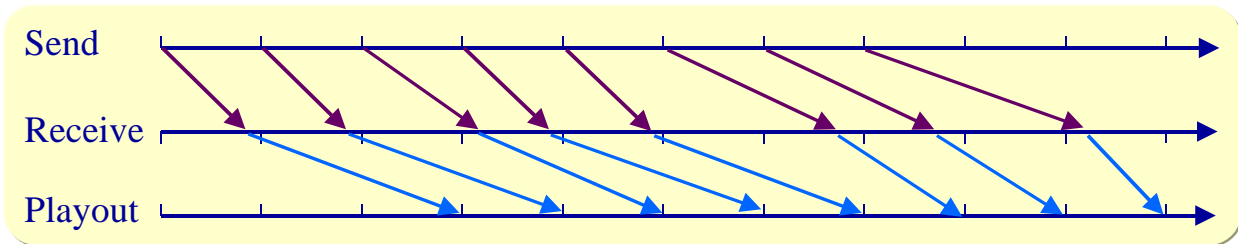
Principles of Delay-Jitter Buffering

Two fundamental initial playout strategies

- ◆ Let naturally occurring network delays determine playout latency



- ◆ Avoid initial sequence of playout gaps by estimating network delay and setting playout delay accordingly

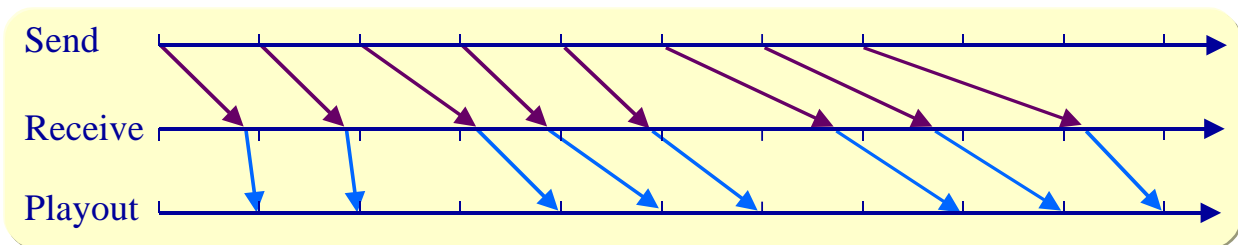


11

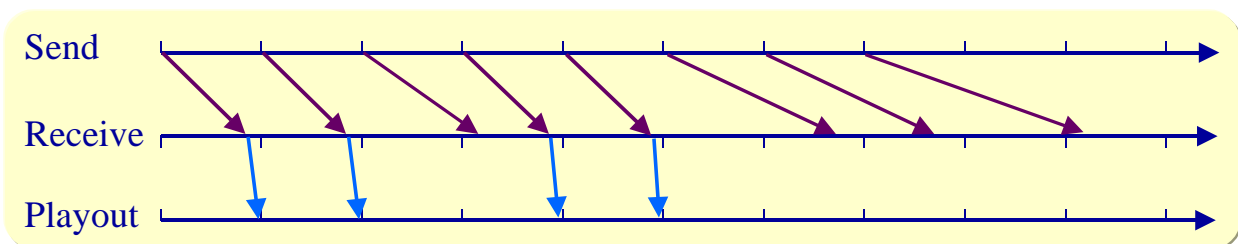
Principles of Delay-Jitter Buffering

Two fundamental late arrival strategies

- ◆ Play media samples as they arrive
 - » Latency increases as delay-jitter increases



- ◆ Discard “late” samples
 - » Playout media with constant latency

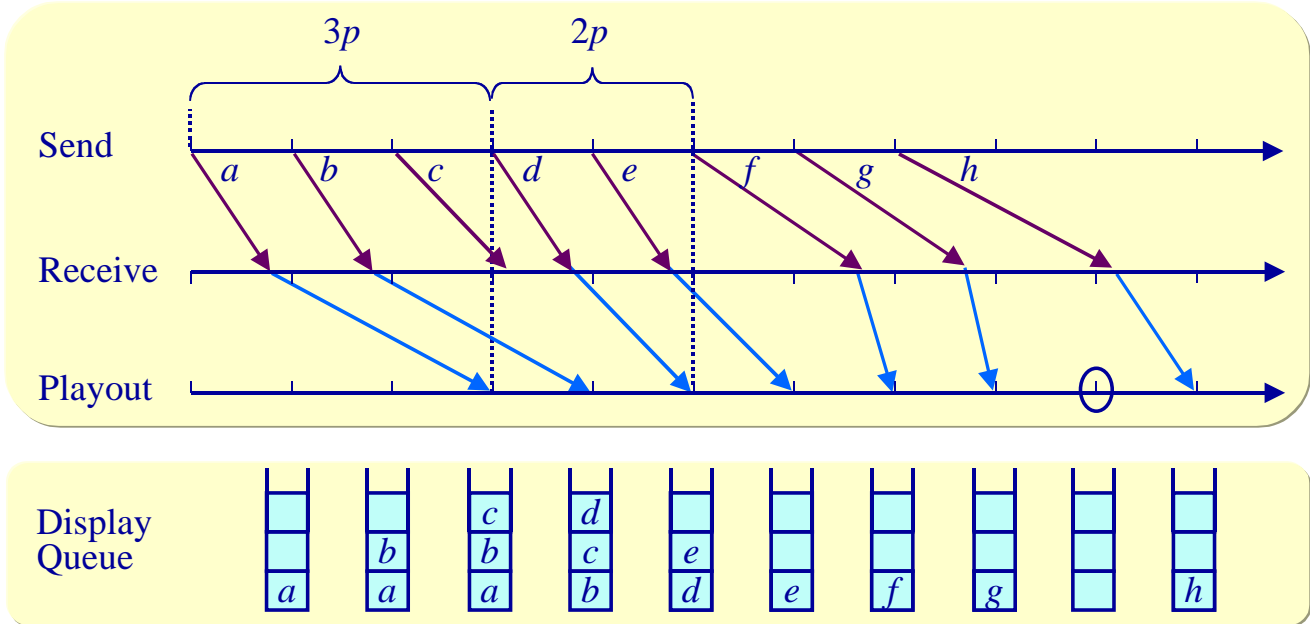


12

Principles of Delay-Jitter Buffering

Two fundamental late arrival strategies

- ◆ More generally, simulating packet loss by discarding samples at the receiver will reduce playout latency



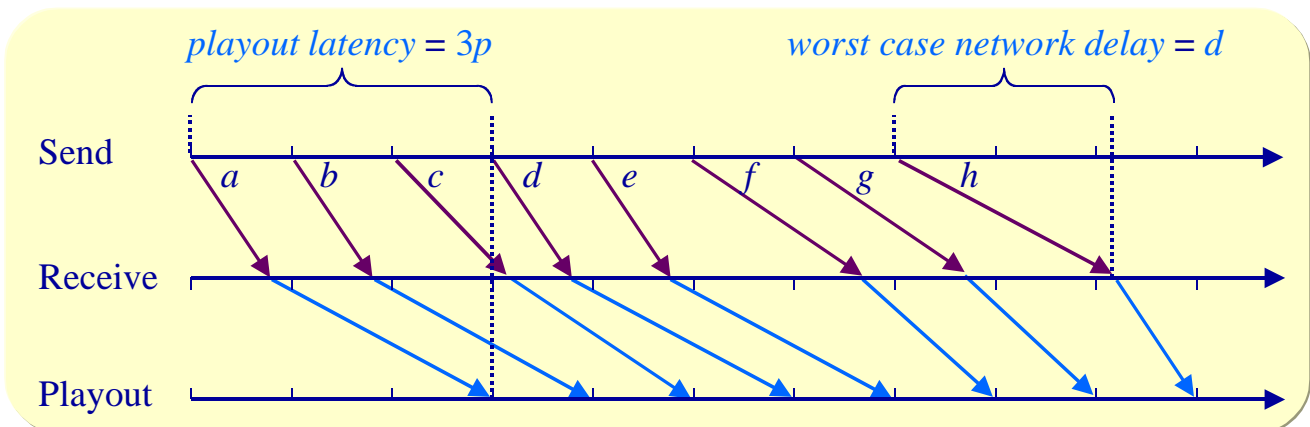
13

Principles of Delay-Jitter Buffering

Estimating network delay

- ◆ If network delay is bounded by a constant d :
 - » Timestamp each packet at sender
 - » When a packet arrives, enqueue the packet and dequeue at time:

$$\left\lceil \frac{\text{sender's transmission time} + d}{\text{media packaging period}} \right\rceil \times \text{media packaging period}$$

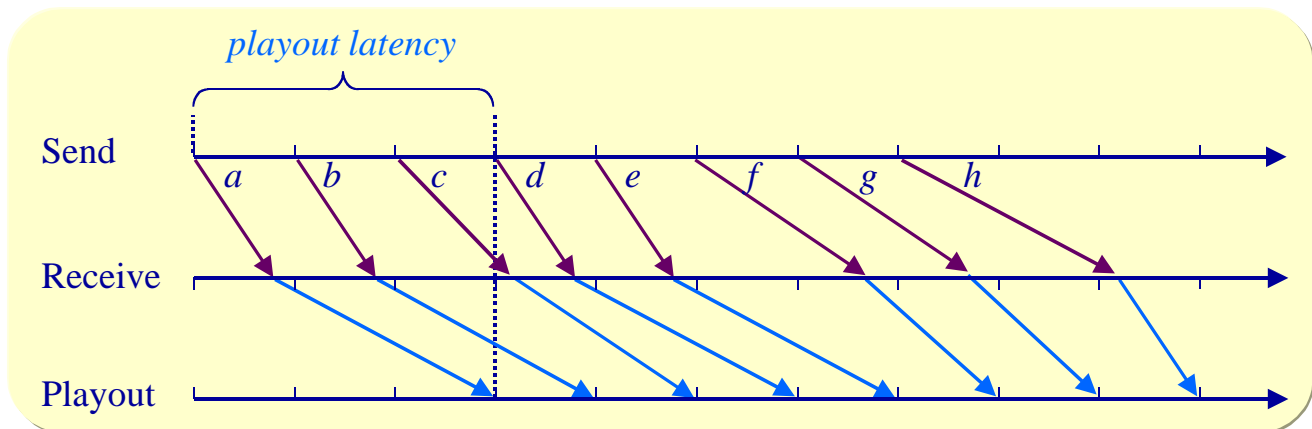


14

Principles of Delay-Jitter Buffering

Estimating network delay

- ◆ Basic algorithm: $playout\ latency \approx d + (k \times v)$, where
 - » d is the average estimated network delay
 - » v is the estimated variation deviation
 - » k is a “congestion estimator”



15

Principles of Delay-Jitter Buffering

Estimating network delay

$$playout\ latency = d + (k \times v)$$

- ◆ The average network delay and variation can be estimated by:

$$d_{new\ estimate} = d_{old\ estimate} + \alpha \times (d_{observed} - d_{old\ estimate})$$

$$v_{new\ estimate} = v_{old\ estimate} + \beta \times (|d_{observed} - d_{old\ estimate}| - v_{old\ estimate})$$

OR

$$d_{new\ estimate} = \alpha d_{old\ estimate} + (1 - \alpha) d_{observed}$$

$$v_{new\ estimate} = \beta v_{old\ estimate} + (1 - \beta) \times (|d_{observed} - d_{old\ estimate}|)$$

OR

$$d_{new\ estimate} = \text{MIN}(d_{observed} \text{ in the recent past})$$

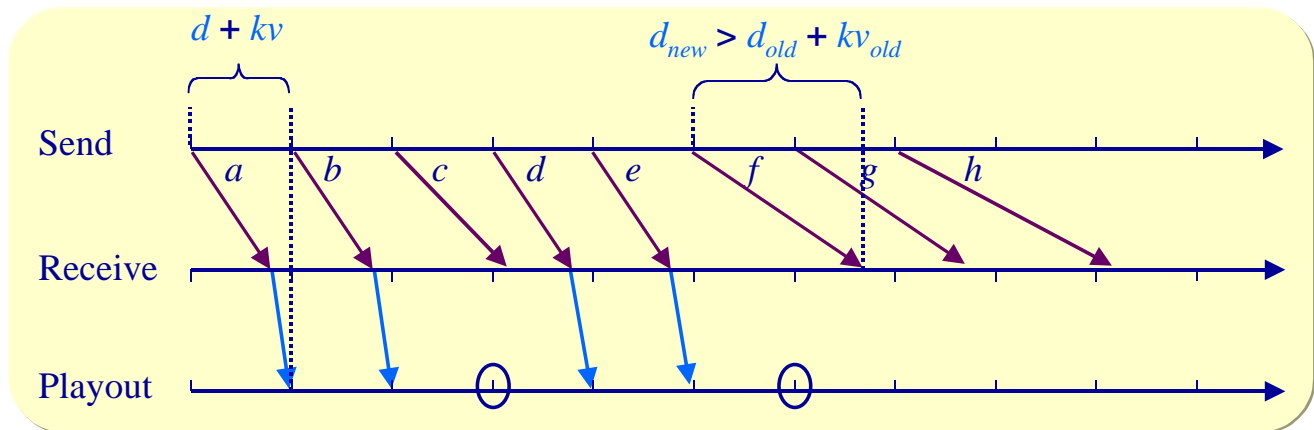
16

Principles of Delay-Jitter Buffering

Estimating network delay

- ◆ All samples are scheduled for playout at time

$$\left\lceil \frac{\text{sample generation time} + d + (k \times v)}{\text{media packaging period}} \right\rceil \times \text{media packaging period}$$

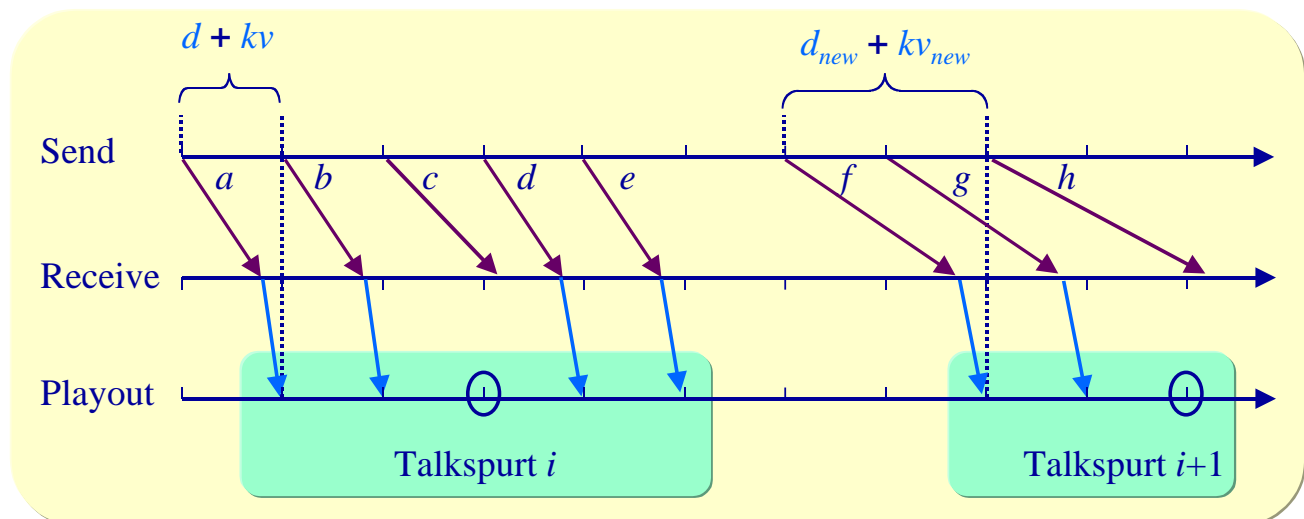


17

Adjusting Playout Latency

Voice transmission

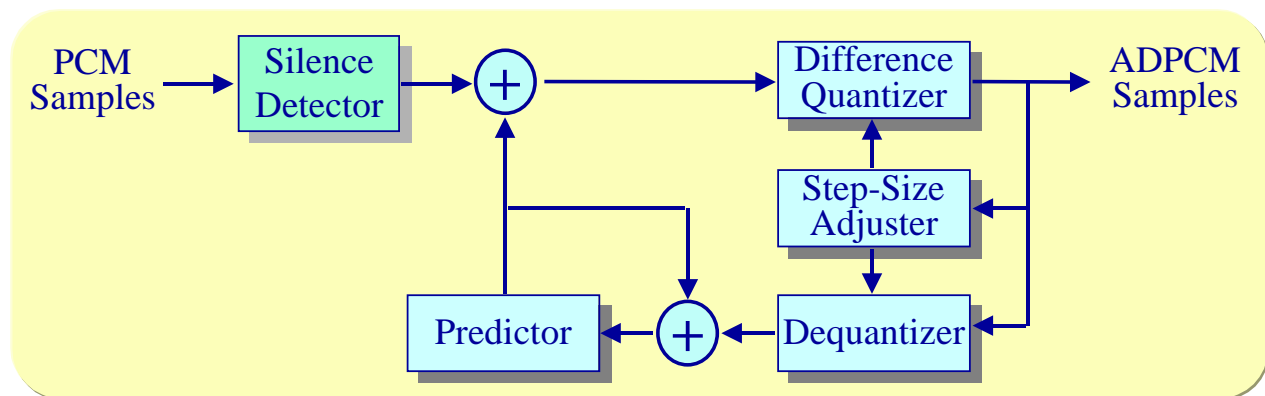
- ◆ For voice transmission we can dynamically adapt playout times of audio samples using silent periods to “resync” the stream



18

Voice Transmission

Silence detection



- ◆ Silence detection is the process of search for a sequence of consecutive audio samples that have the property

$$|sample_i - sample_{i-1}| \leq threshold$$

19

Adjusting Playout Latency

Continuous audio transmission

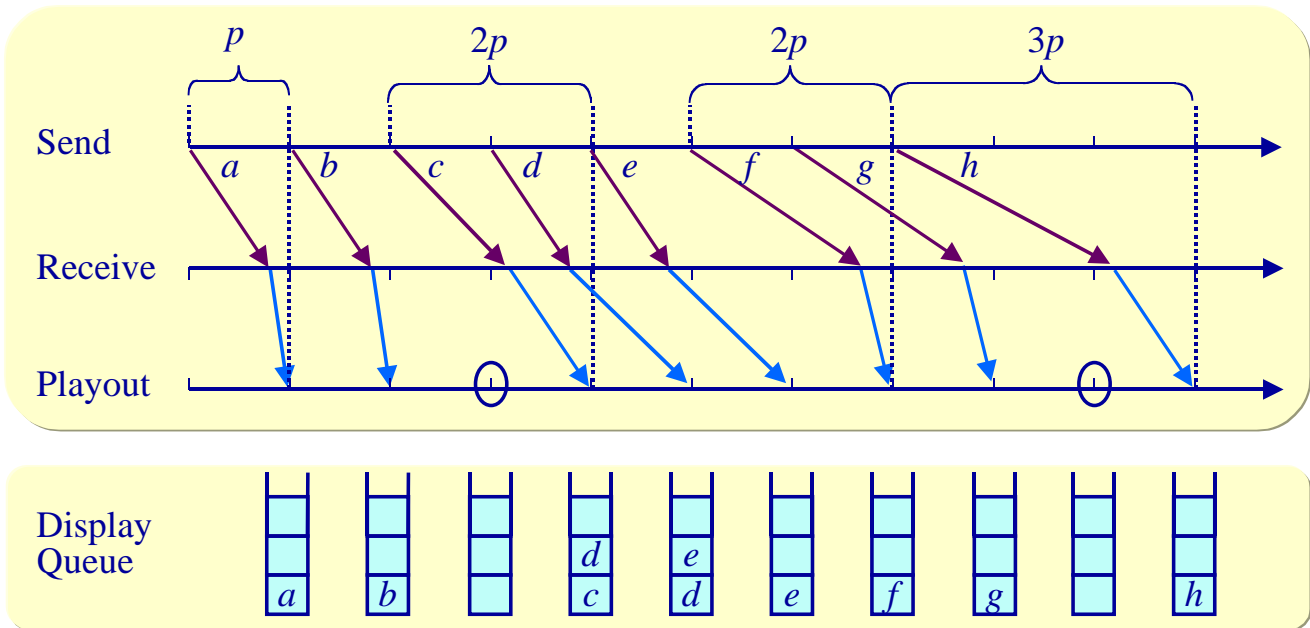
- ◆ Scheduling individual samples for playout based on estimates of network delay can give poor results
 - » Delay-jitter is an end-to-end measurement
 - » Should we rely on the existence of synchronized clocks?
- ◆ Many forms of audio (and other media) must be transmitted continuously
 - » Music
 - » “Noisy” voice
 - » Mixed audio streams
 - » Video?

20

Adjusting Playout Latency

Continuous audio transmission

- ◆ Let naturally occurring network delays determine playout latency

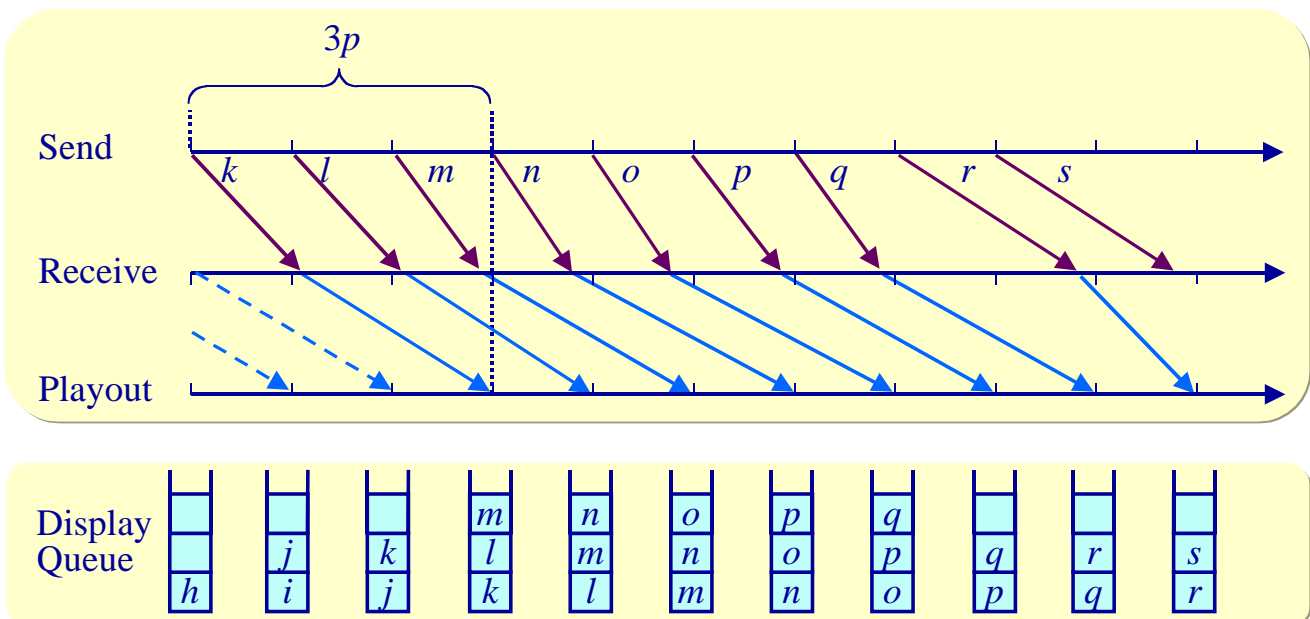


21

Adjusting Playout Latency

Continuous audio transmission

- ◆ How do we determine if our delay-jitter buffer is too large?

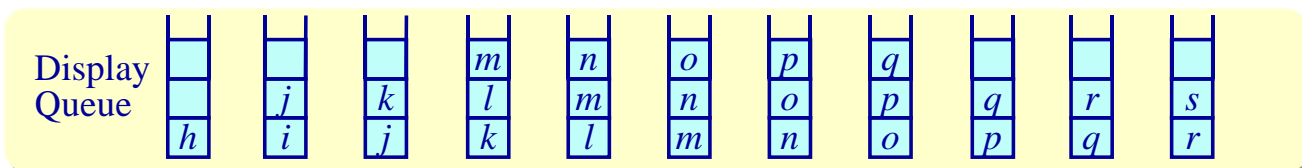
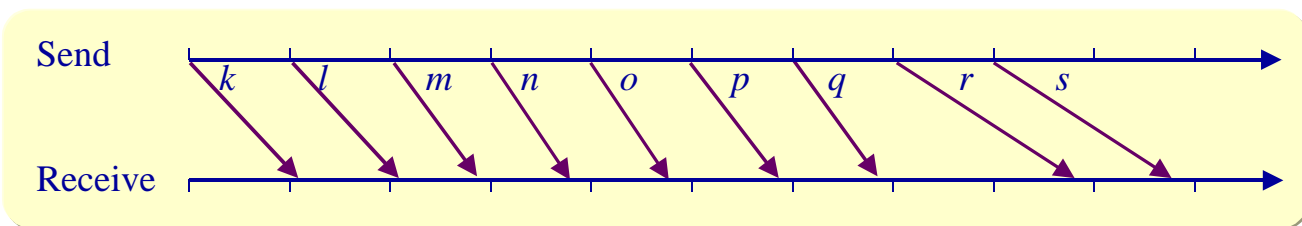


22

Continuous Audio Transmission

Queue monitoring

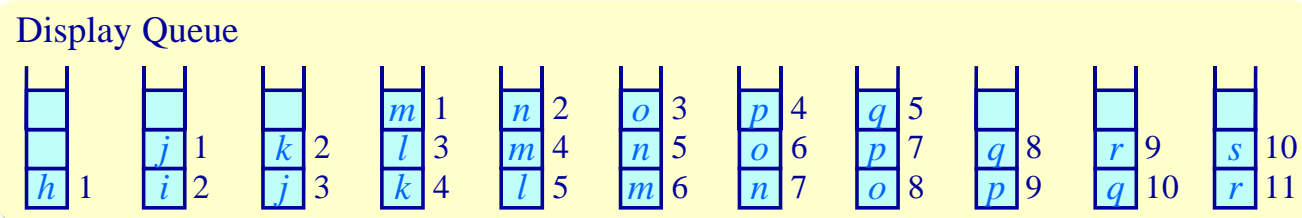
- ◆ Rather than compute network delay, infer it from the length of the display queue
 - » If queue length grows, network delay is decreasing
 - » If queue length shrinks, network delay is increasing
 - » If queue length remains constant, network delay is stable



23

Queue Monitoring

Implementation



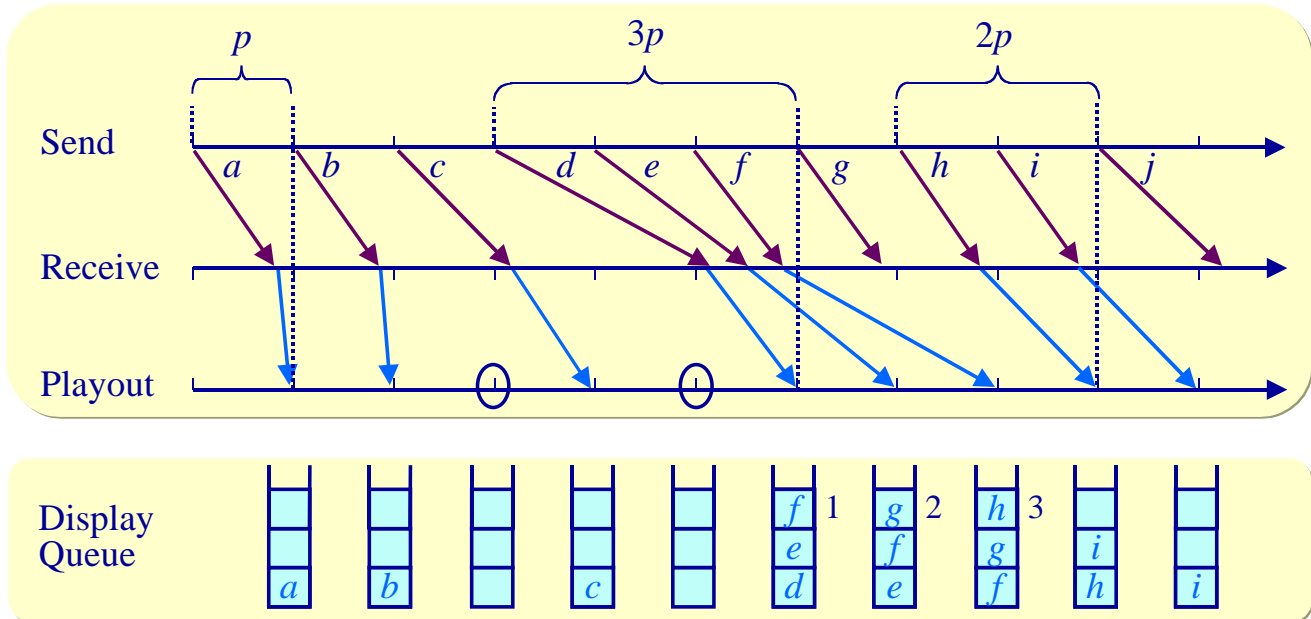
- ◆ Keep count of the number of consecutive display initiation times at which the display queue contained n items
- ◆ When the count exceeds a threshold, the oldest sample in the queue is discarded
 - » Queue locations near the head of the queue have large thresholds
 - » Queue locations near the tail of the queue have small thresholds

24

Queue Monitoring

Implementation

- ◆ Example: Queue monitoring with thresholds = 3, 10
 - » Sample *g* discarded at playout time 9



25

Queue Monitoring

Performance on a campus-area network

- ◆ How much delay-jitter can be accommodated in practice?
 - » What ranges of delay-jitter are observed?
 - » How well do these buffering schemes work in practice?
 - ◆ Stone's delay-jitter study in the UNC CS department:
 - » A comparison of the effectiveness of three delay-jitter management policies:
 - I-Policy — playout media with fixed latency
 - E-Policy — playout media samples as they arrive
 - Queue Monitoring — adaptively set the playout delay
- on the playout of audio/video in a videoconferencing system

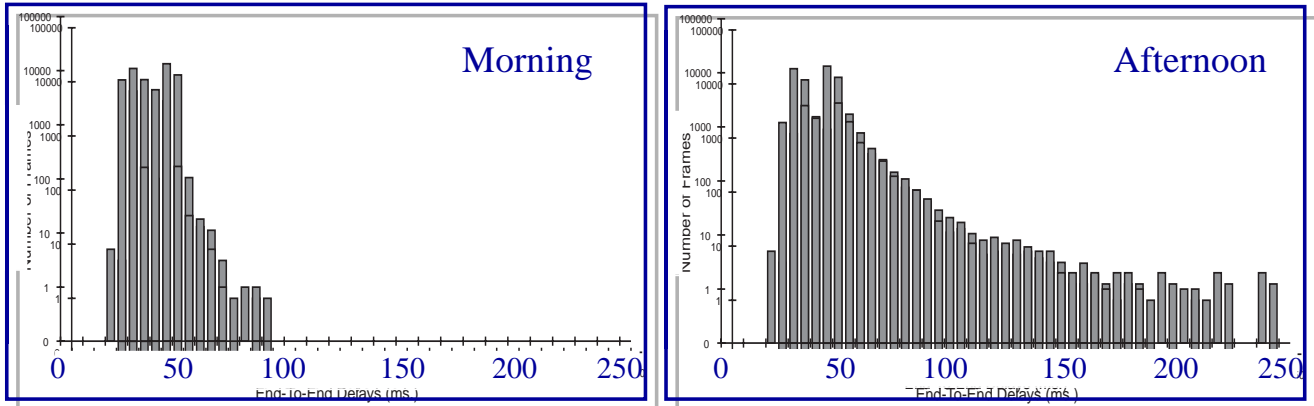
26

Queue Monitoring

Performance on a campus-area network

◆ What ranges of delay-jitter are observed?

» Stone measured the performance of 28, 5 minute conferences during the course of a “typical” day

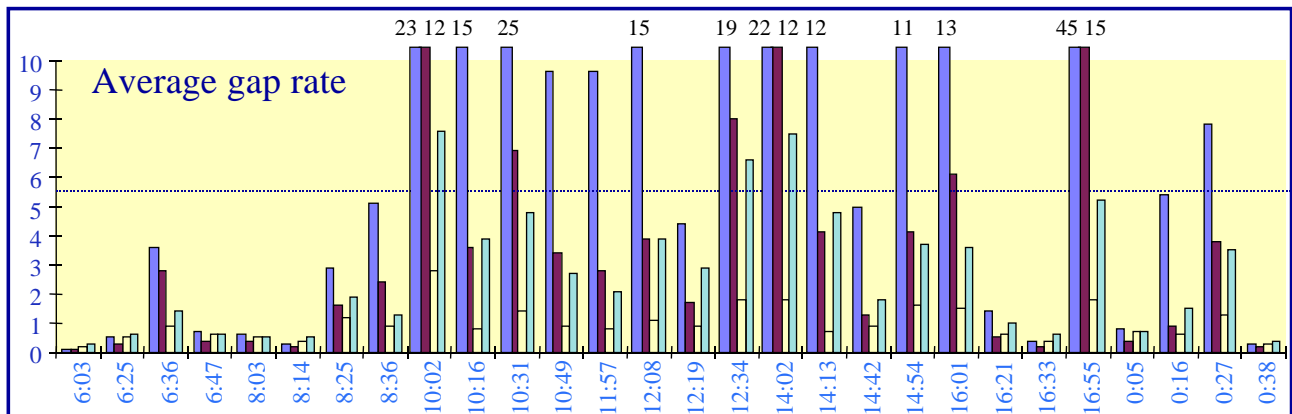
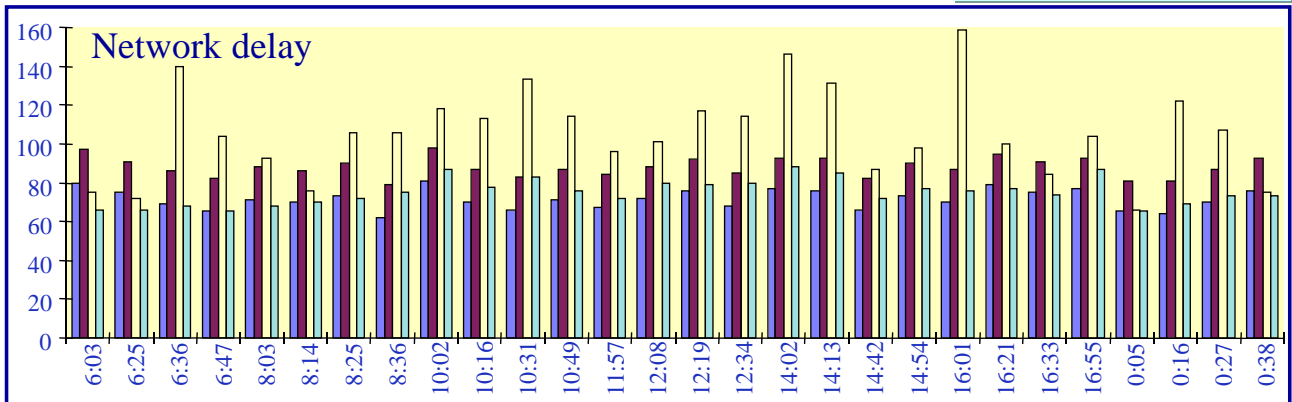


Audio end-to-end delay distribution (in ms)

Queue Monitoring

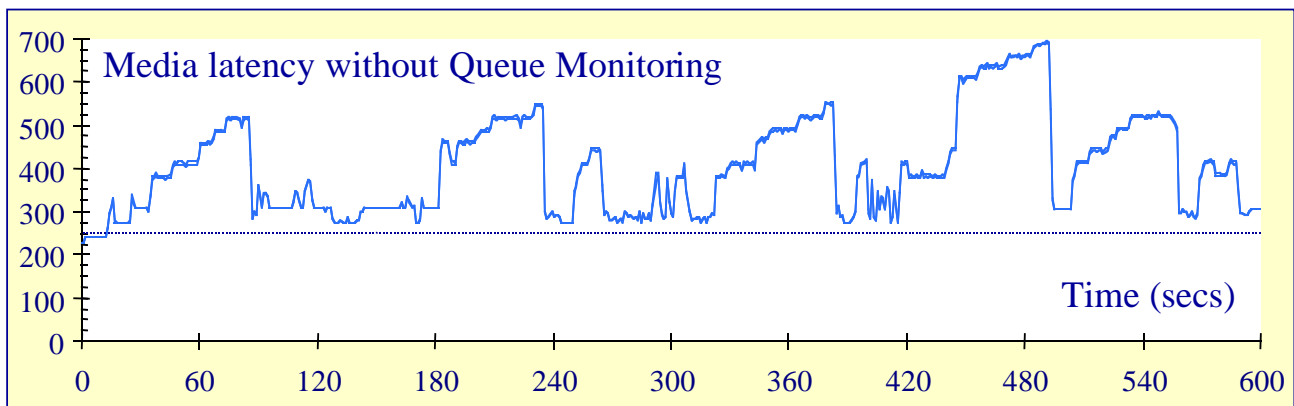
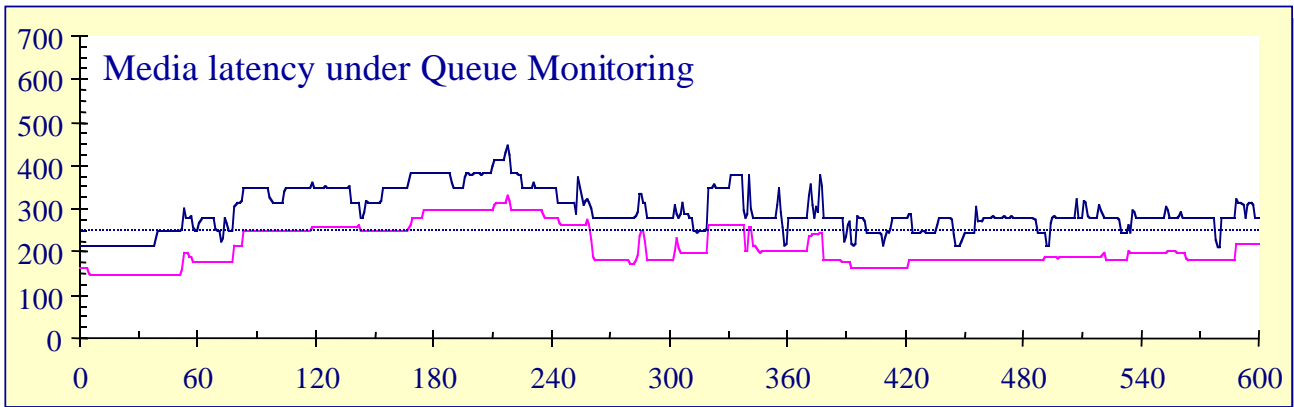
Performance on a campus-area network

- I-2 Policy
- I-3 Policy
- E Policy
- Queue Monitoring



Queue Monitoring

Performance on a campus-area network



Principles of Delay-Jitter Buffering

Non-real-time media transmission



- ◆ If the communication is non-real-time, doesn't simple static buffering solve the problem?
 - » Yes, but...