Applying ALF: Some aspects of reliability in lightweight sessions on the Mbone

Mark Handley

currently at:

Department of Computer Science

University College London

M.Handley@cs.ucl.ac.uk

How this talk fits together...

...if it does at all...

Background

On packet loss and multicast based conferencing.

Why I'm not talking about video.

Audio

The RAT tool, human perception of packet loss in audio, what we can do to minimise the effects of loss.

Shared Tools

The Nt shared editor, how we use shared tools, what we can get away with and what we can't, the interplay between redundancy and retransmission mechanisms.

Session Directory

The Sdr session directory and how it's evolving to be more reliable than sd.

Common themes

you mean there are some?

Background: Packet Loss

Packet loss comes from a number of sources:

- **Congestive Loss**
  - **transient in properly provisioned networks**
  - **persistent in under-provisioned networks**
  - **packet losses are effectively independent if...**
- **Burst Loss**
  - **so-called “Cisco Disease”**
  - **routing changes**
- **Partitionings**
  - **link/router failures**
  - **routing errors**

Background: General Aims

- ***survive*** **partitionings**
- ***prevent*** **burst loss**
- ***adapt*** **to reduce congestive loss where possible**

  or

  tolerate **it where not.**

This talk is mostly about tolerating loss and surviving partitionings.

Assumptions

- **IP multicast for efficient data distribution and as a rendezvous mechanism.**
- **A significant number (>100) of users participate in any session we are designing for.**
  - **if possible mechanisms should cope with very large numbers (thousands to millions) of users, though this may not always be appropriate for all tools.**
- **The network is heterogeneous, but that in many (most?) places our traffic is a small fraction of the total traffic.**
  - **fast backbones, slow subscriber loops, etc**
  - **both transient and persistent congestion may occur in multiple places in the multicast tree.**
  - **probability of any particular packet being lost by at least one receiver is significant.**

What about reservations?

They should help, but...

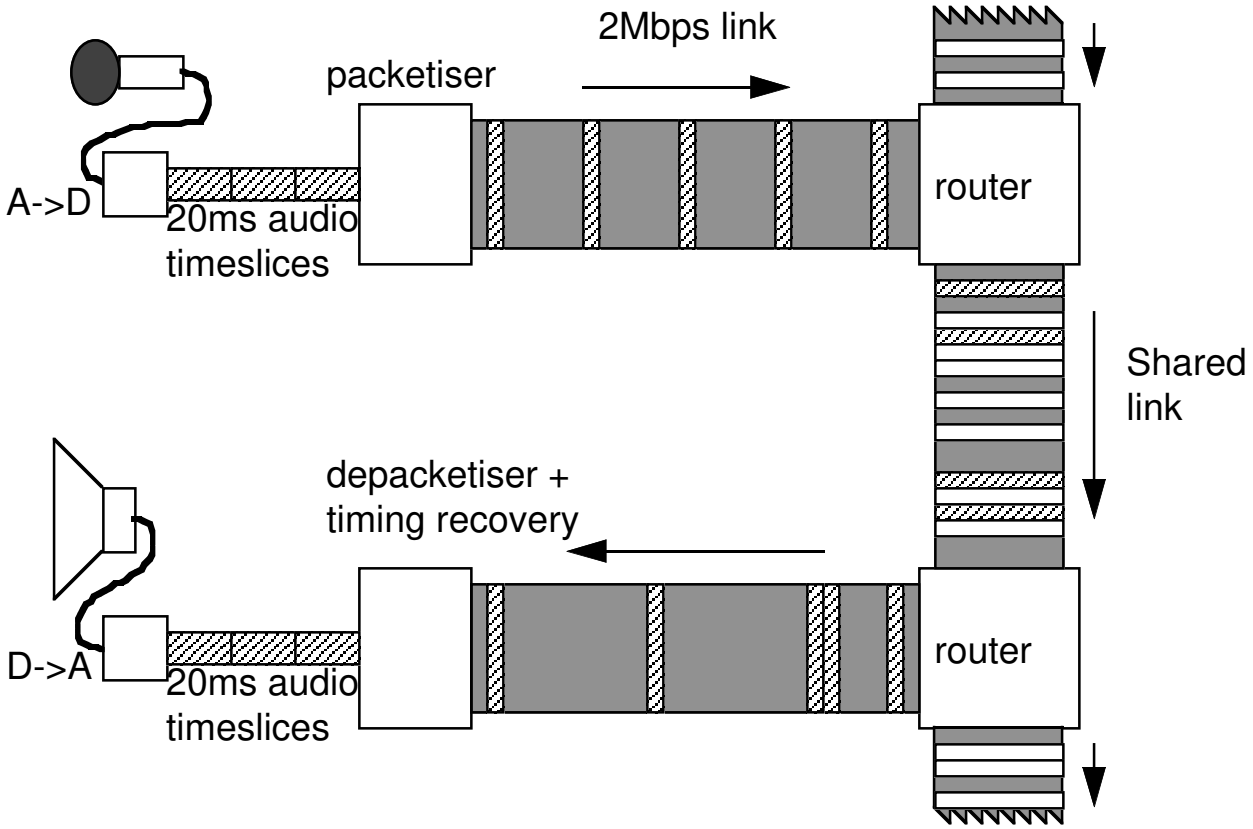
...lets see how far we can get without them...

(am I allowed to say that here?)

Audio: Acknowledgements

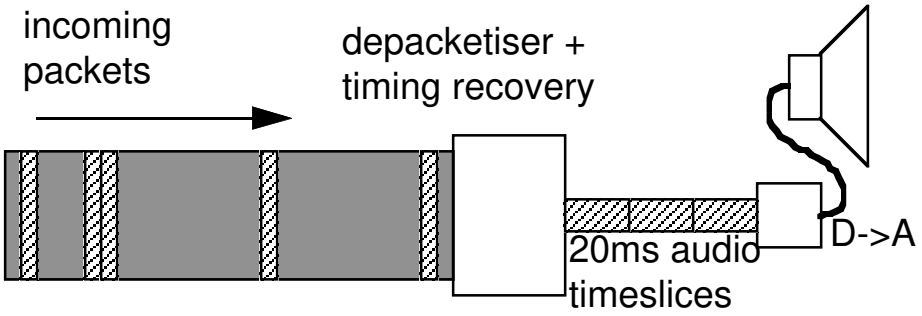
- Most of this work was performed by and is implemented in the RAT audio tool by Vicky Hardman and Isidor Kouvelas of UCL, and by Jean Bolot at INRIA.
- See `http://www.cs.ucl.ac.uk/mice/rat/` for more details.

Packetisation and Jitter

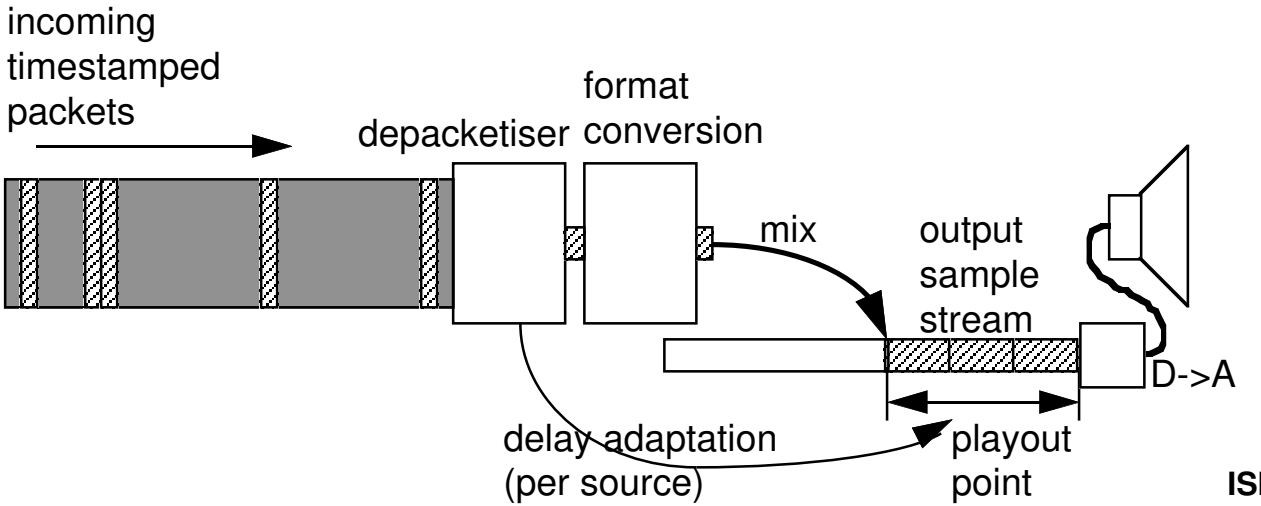


- jitter is caused by some packets being delayed for longer than others.
- packets can also be lost or re-ordered.

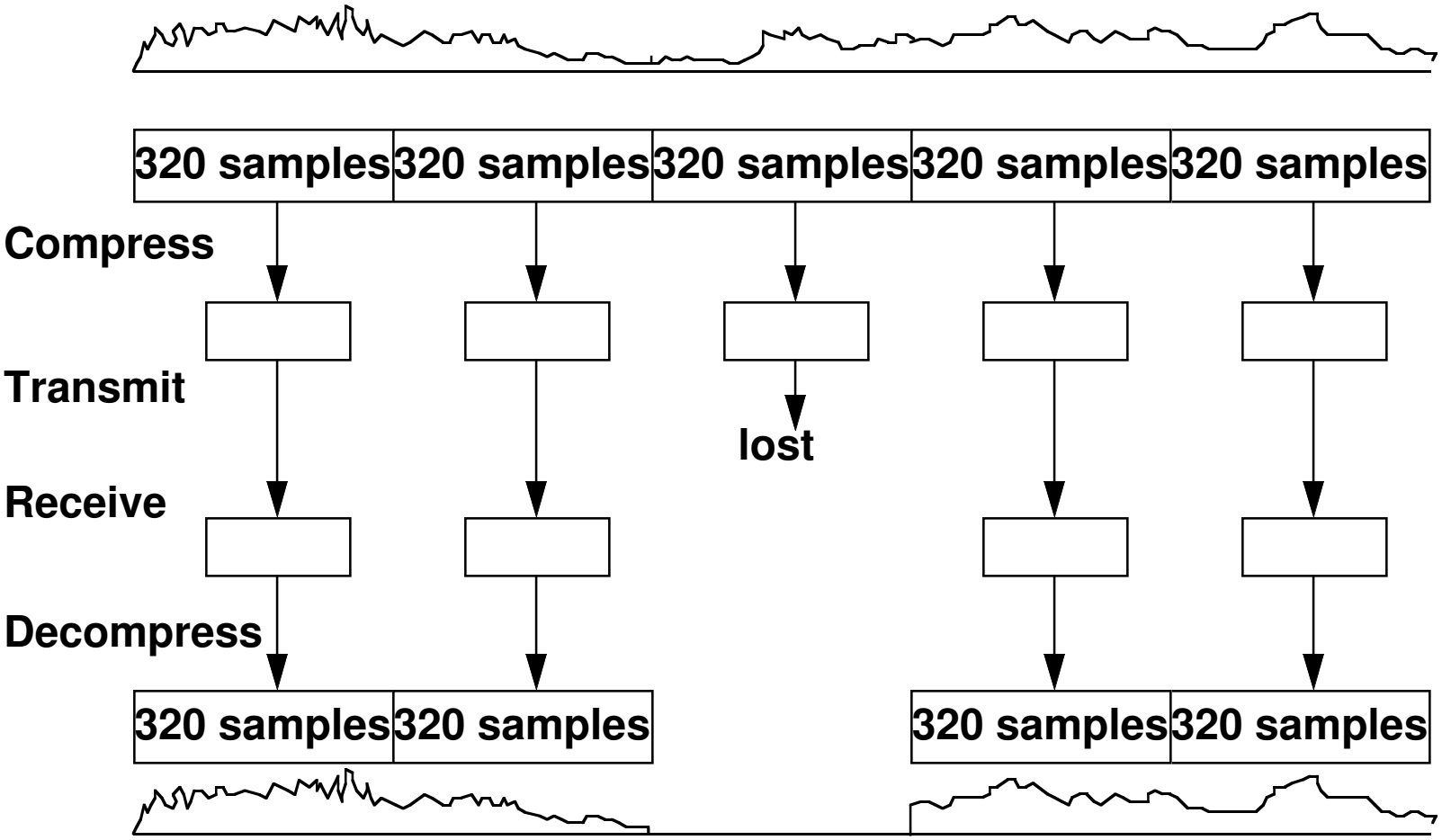
Adaptive Playout Buffers



so our adaptive receiver now looks like:



Audio Packet Loss



Effects of Packet Loss

- Phonemes are approximately 10-40ms in length.
- Packet header overheads dictate that we choose >20ms per packet.
- Interactivity requires <=80ms per packet
  - end-to-end delay = (80ms delay at sender + network transit delay + playout buffer delay + device jitter delay)
  - absolute maximum round-trip delay should be less than 500ms for interactivity and less that 100ms to be unnoticable.
- Problem: phonemes are roughly the same length as packets.
  - losing whole phonemes significantly affects understandability.

Options for dealing with loss

- replace lost samples with silence
  - turns out to be about the worst you can do because the brain treats silence as significant!
- replace lost samples with noise
  - the brain is used to dealing with background noises so doesn't treat them as significant.
  - information is still lost.
- replace lost samples with something resembling what was lost
  - packet repetition
    - + assumes speech doesn't change quickly
  - redundancy
    - + recreate lost data (FEC)
    - + recreate approximation to lost data

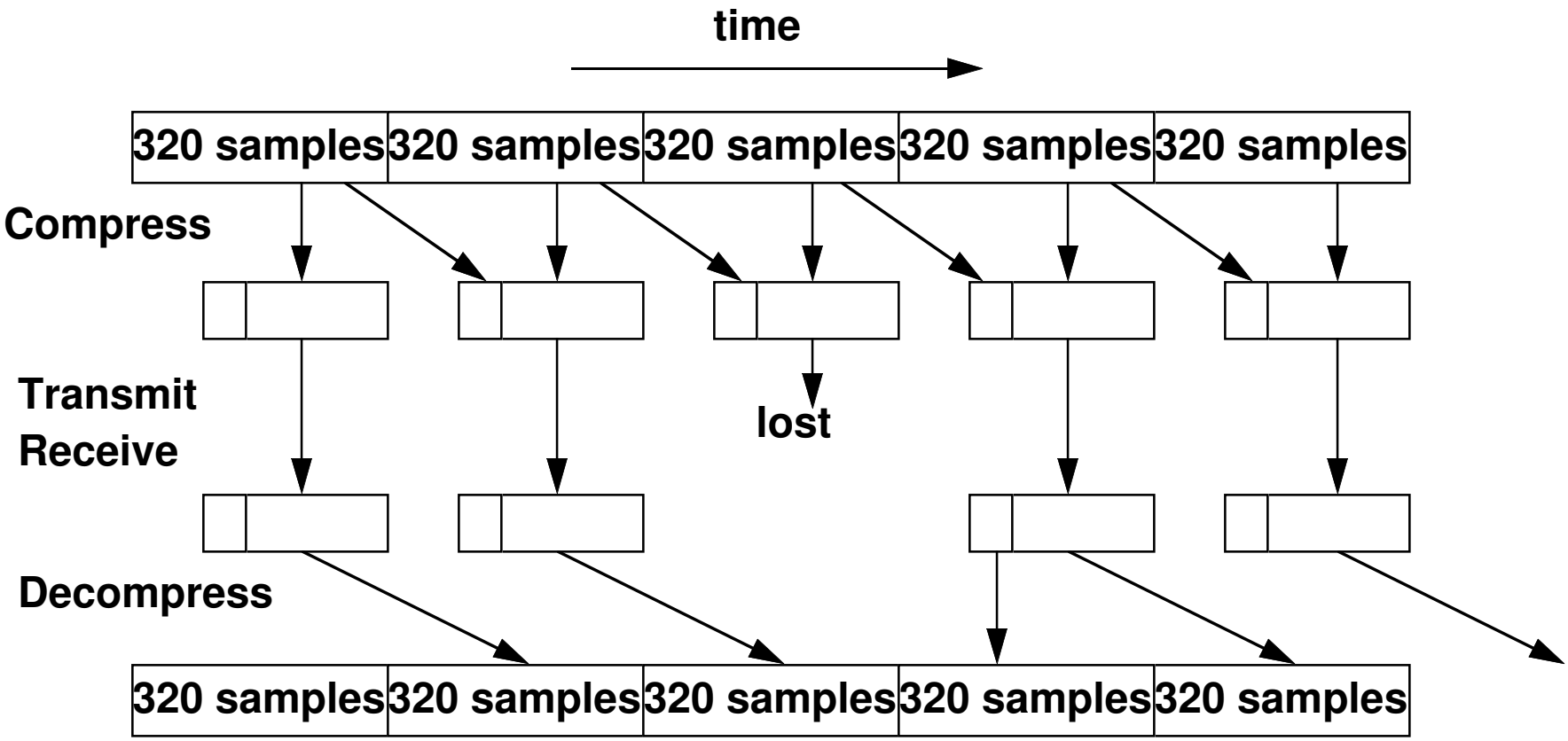
Constraints

Delay, delay, delay!

Bandwidth too...

- **Packet repetition is effective with shorter packet sizes.**
  - **Might be even more effective if short time intervals are alternated between two packets, but this adds a one packet delay and also spreads loss.**
- **Packet-level FEC generally adds too much delay for interactivity. Significant extra b/w.**
- **Adding a low rate secondary compression scheme can be very effective.**

Redundant Encodings



Adaptation

- The general idea is that in the presence of loss, sending one high quality stream is not the best use of the available bandwidth.
- Sending a lower quality primary stream and a very low quality redundant secondary stream will result in better perceived quality.
- The sender should adapt the redundancy level depending on receiver reports.
- Receivers with a significant loss rate can adapt and trade delay for quality by increasing their playout point to be able to decode the secondary scheme.

Experimental Analysis

- **Anna Watson at UCL conducted an extensive set of speech intelligibility experiments with real subjects.**
- **A very brief summary:**
  - **silence substitution was unintelligible above 10% loss and unpleasant at lower loss rates.**
  - **packet repetition extended the intelligibility point significantly for 20ms and 40ms packets but worked less well for 80ms packets.**
  - **4Kbps LPC redundancy worked well up to 20-25% loss and worked well with 80ms packets.**
  - **The brain does not treat the LPC-ADPCM transitions as significant artifacts (something we had wondered about).**

Future Options

- **Intelligent packetisation based on speech analysis.**
- **Redundancy based on speech analysis**
  - **for example, the first phoneme of a work is more important in English than the rest.**

Nt - a shared text editor for the Mbone

ftp://cs.ucl.ac.uk/mice/nt/

- Van Jacobson's whiteboard wb is a very effective whiteboard, but it does not permit collaboration over text as it does not permit users to edit each others objects.
- Under some high loss situations, wb becomes unusable due loss of its single-character packets and due to it rate-limiting its retransmissions.
- Interactive collaboration over an "object" such as a line of text requires very timely loss repair.
- These motivated the design of nt.

User Requirements

1 - many people should be able to manipulate the same data object over time.

2 - eventual consistency

- everyone should end up with the same dataset

3 - deterministic behaviour

- if I typed it I want it to stay typed...

4 - fully interactive

- I don't want to have to wait for locks or such like

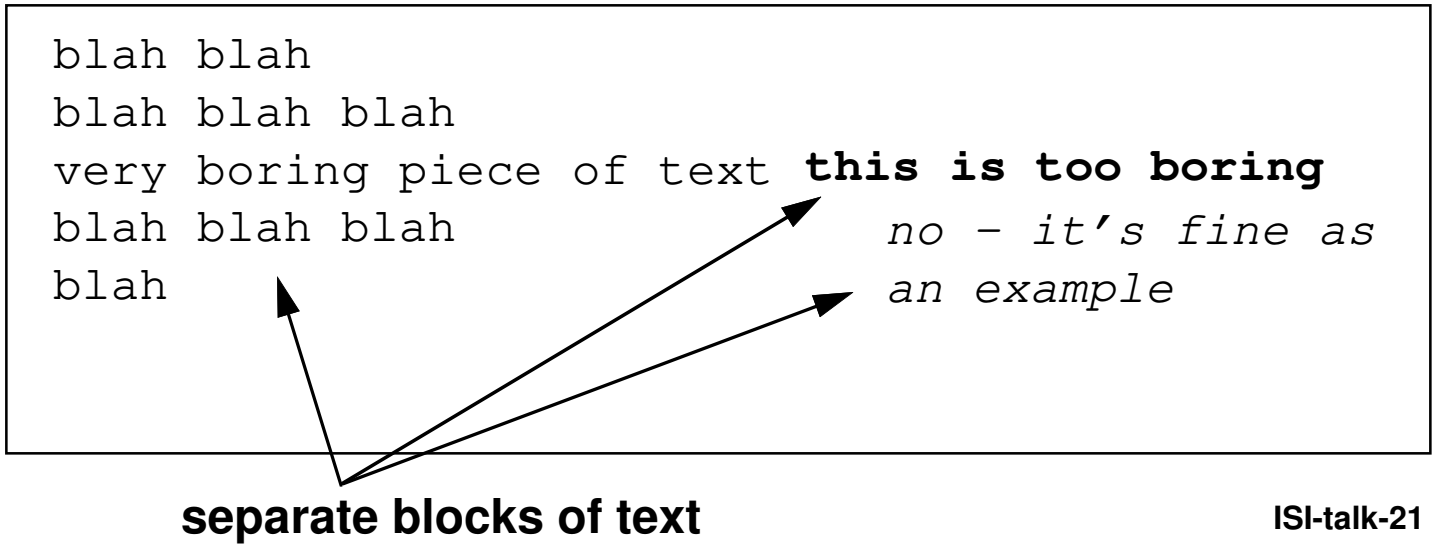
The problem is that these requirements are contradictory given unreliable networks.

Realistic User Requirements

- **Many people should be able to manipulate the same data object over time.**
  - **wb doesn't allow this, but we must do for a shared editor**
- **Eventual consistency**
  - **probably can't compromise on this**
- **Fully interactive**
  - **most systems compromise this in the presence of loss**
- **Deterministic behaviour**
  - **if I typed it I want it to stay typed... *unless*....**
  - ***what happens if we relax this requirement a little?***

Nt design

Collaboration tends to dictate an editor that resembles a drawing package in that different users want to be able to work simultaneously and to add comments or annotations that are separate from the text being worked on.



Nt design

block and line based

↓

line is our ADU

↓

send whole line for each change

↓

gives great redundancy

for little additional overhead

↓

but now how do we do reliability?

Redundancy and reliability

- if you keep typing most loss is naturally corrected, and the meaning of what you're saying is conveyed without the need for any retransmissions
  - thus we don't need (or want) all lost packets to be retransmitted if they'll be naturally replaced
  - we don't want the retransmission scheme to be too fast
  - we do want it to be bounded and predictable in the time it takes
  - most importantly, we do want eventual consistency

General Idea

timestamp each change to a line and send the whole line, line-id and timestamp each time the line is changed.

if an incoming object has a later timestamp than our current version, the incoming version is correct.

needs 3 mechanisms:

- clock synchronisation
- an announcement mechanism
- a nack-implosion avoidance mechanism

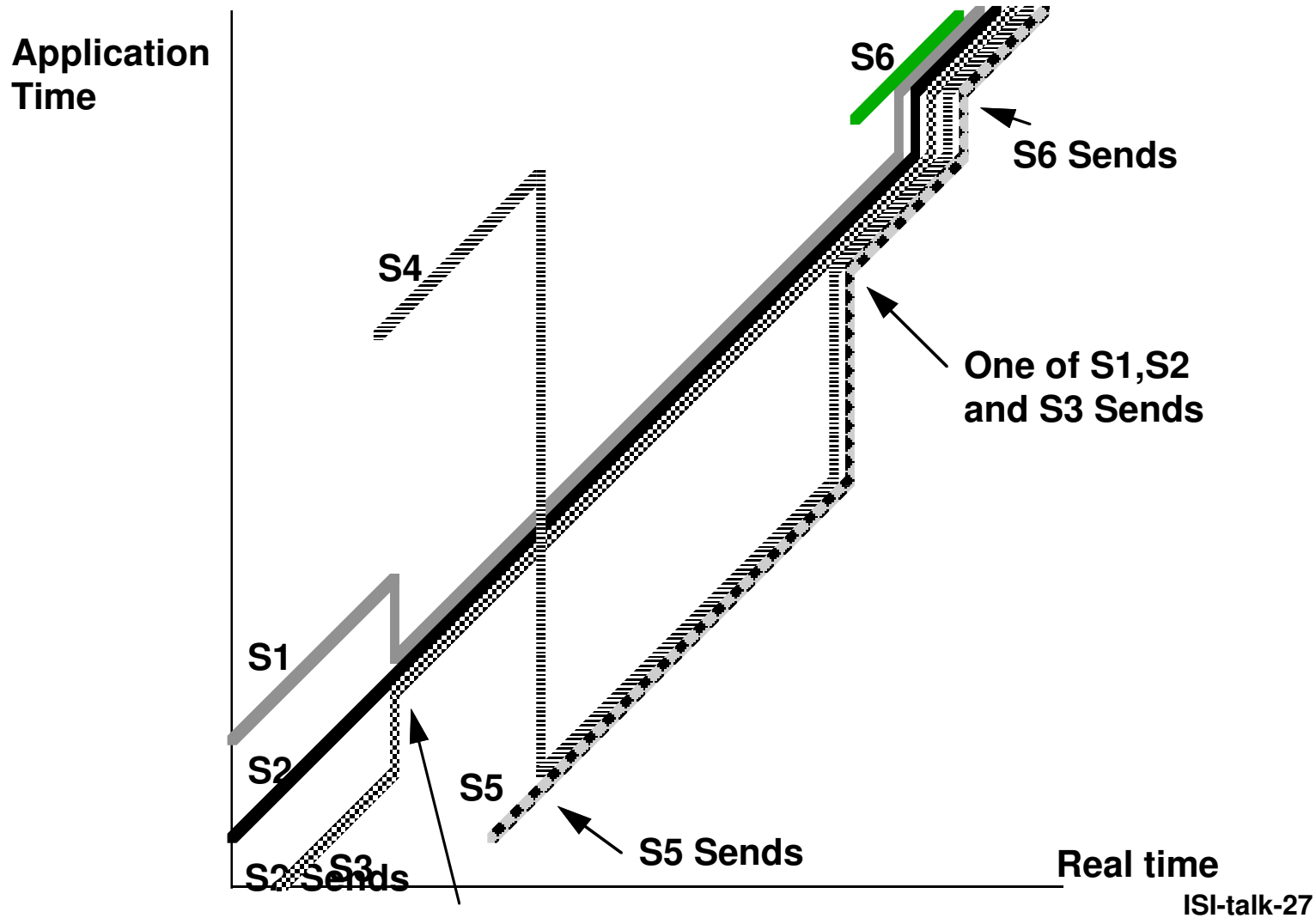
Clock Synchronisation

- For our purposes, the basic requirements are that a clock never moves backward due to any adjustment, and that all clocks converge on some (arbitrary) time.
- Given that all changes are multicast, and that all packets contain timestamps, we have a simple mechanism for clock synchronisation

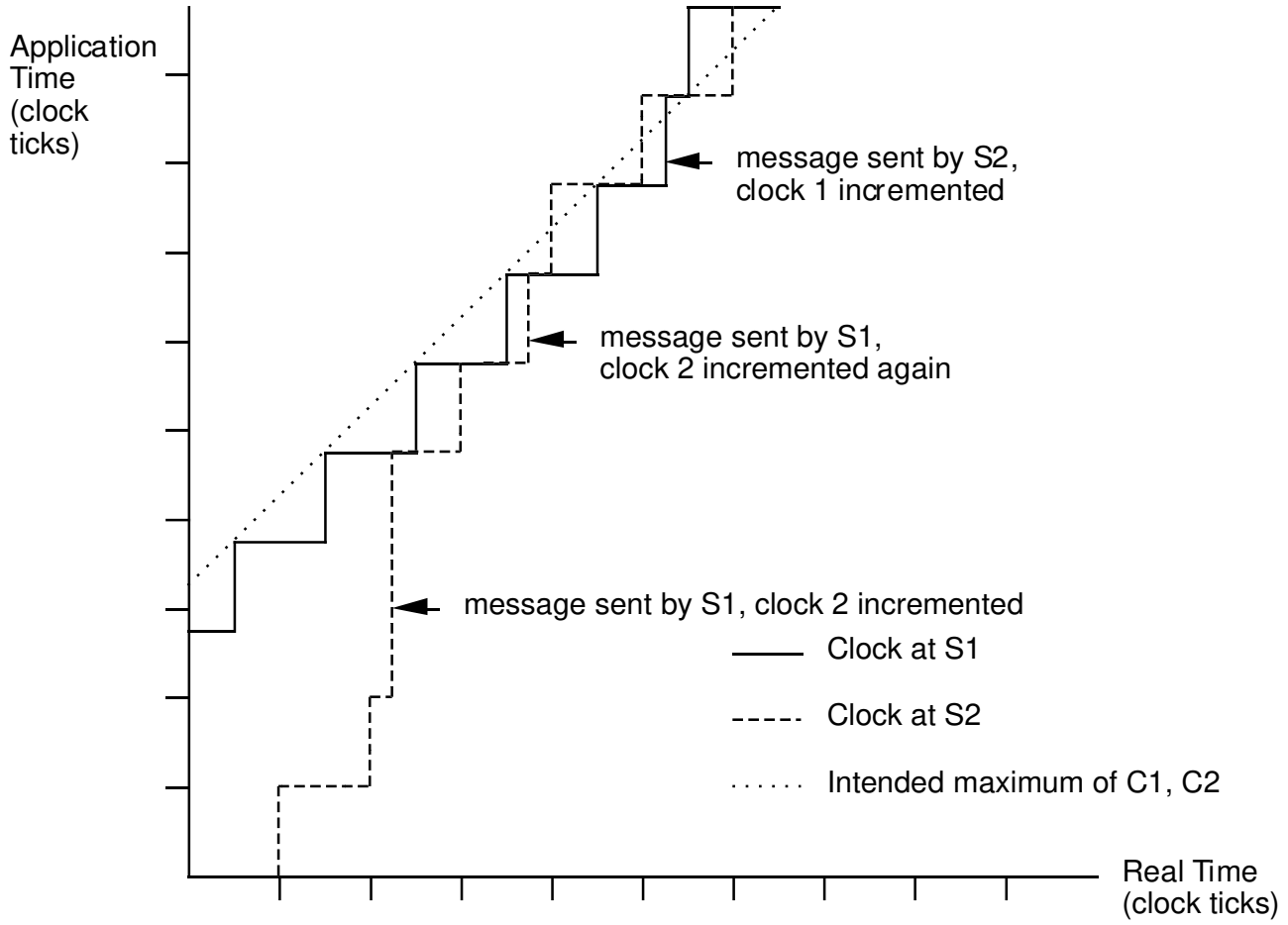
Clock Sync

- if a site has not sent any data and receives data, it sets its app clock to the timestamp in the packet.
- if a site has not received any data and needs to send data, it sets its app clock to its own real-time clock.
- if a site then receives a message with a timestamp greater than its current app clock time, it increases its app clock to equal that in the received message.

Clock sync


Application Time
S6
S6 Sends
S4
One of S1,S2 and S3 Sends
S1
S2
S5
S5 Sends
S2 Sends
S3
Real time

Clock Granularity Problem



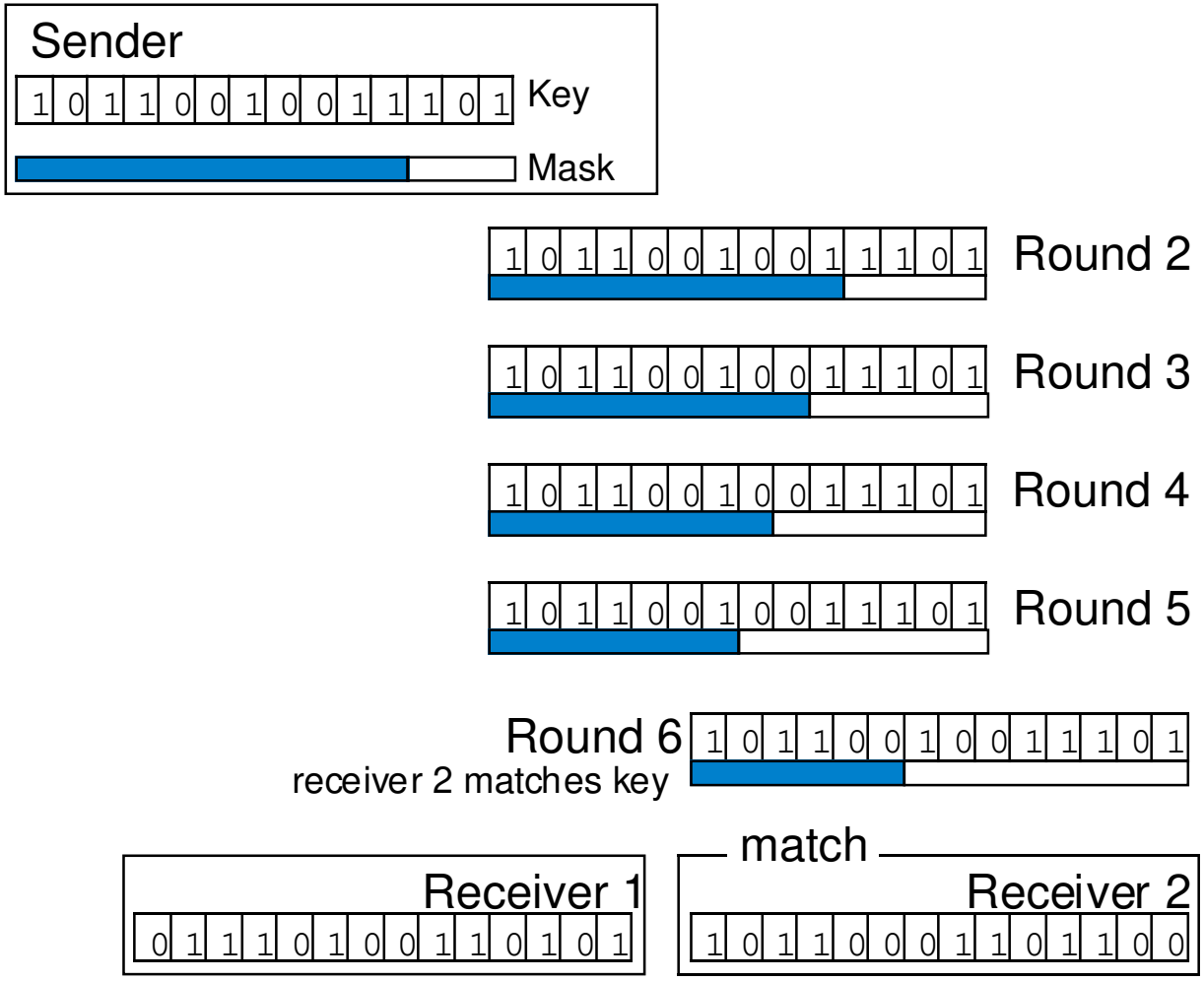
Announcement Mechanism

- This is the mechanism by which sites discover that they have an out-of-date version of an object or a missing object.
  - though missing objects are usually noticeable by references from other objects.
- Two main situations where we need this:
  - packet loss on the last of a sequences of changes to a line.
  - recovery from network partitioning
- We could distribute this mechanism
  - have sites each list one recently modified object
- We could “centralise” it
  - elect a site to list all the objects that recently changed

Nack-implosion avoidance

- This mechanism may not be optimal for general use, but it pre-dates any description of SRM, it is novel, and it works!
- Also it can be combined with the announcement mechanism.
- It is based on the sliding-key scheme devised by Ian Wakeman (see Sigcomm ‘94) for scalable congestion control for video.

Sliding Keys



Sliding Key Mechanism

- Avoids NACK implosion problem
- O(logN) delay before being able to make a request
  - depending on whether others got loss too.
- Simple to implement
  - requires minimal tuning to work well.
  - SRM requires much more tuning and RTT matrix estimation.
- Works well when combined with an announcement mechanism
- Would I do it this way now?
  - Perhaps. Possibly I'd use an SRM based scheme...

There's much more...

- Checksum mechanism for solving long-term partitioning.
- Constraints on line generation and deletion.
  - basically, ordering of lines in a block can only be modified by creation of new lines.
  - deleted lines are left in place but no longer visible.
  - this allows many inconsistencies to be automatically corrected and the rest to be detected and then corrected by an explicit mechanism.
- User-locking of blocks
  - constrained to ensure deterministic behaviour

Lessons

- Nt works well
  - noticeable inconsistencies are rare (even with high loss)
  - human protocols rapidly evolve to avoid conflict
- There are many reliable multicast schemes
  - one shoe doesn’t fit all
- Application specific clock-sync can be simple
  - though being able to assume NTP would be better
- Redundancy greatly complicates reliability!
- ALF applies to time as well as space

Session Directories
(from sd to sdr)

- **Purpose:**
  - Advertise sessions and the information necessary to participate in them to potential participants
  - Multicast Address Allocation
- **Constraints:**
  - Scalability
  - Scalability
  - Scalability
  - Timeliness

Scalability

- Receiver group > sum of all session groups
  - can assume someone lost your packet
  - receiver requested retransmission not a useful mechanism
- Bandwidth used should be a small proportion of the session bandwidth
- Finding relevant sessions is a scalability issue
- Multicast address allocation is a scalability issue

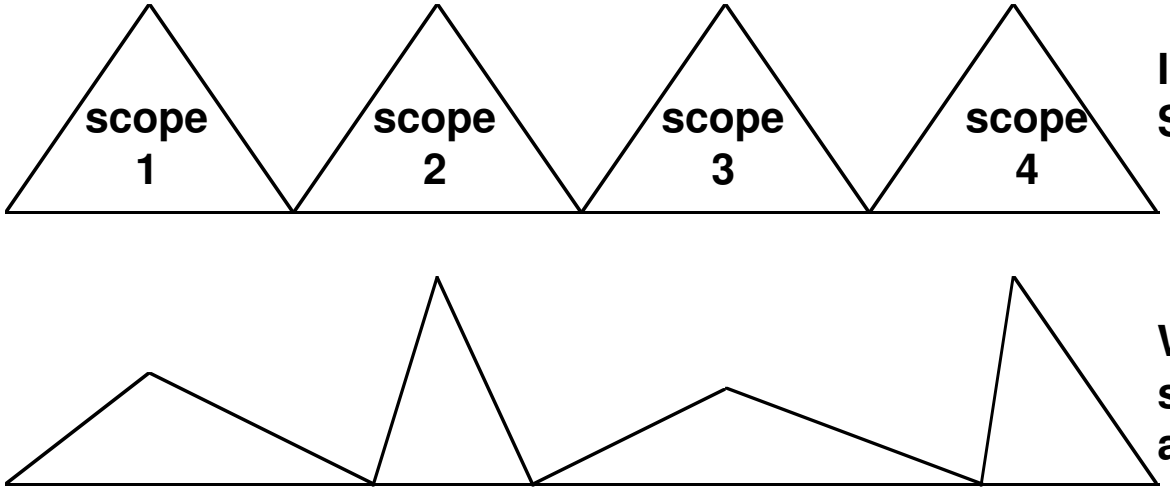
Announcements & Allocation

- Simple periodic multicast works well
- Can rate limit total session announcement rate to keep it to a small constant
  - interval between announcements scales O(N)
- Address allocation may be a limiting factor
  - random based address allocation schemes suffer from the birthday effect
  - number of sessions allocated before a clash is expected scales O(sqrt(N))
  - can do better than random...

IPRMA

Informed Partitioned Random Multicast-address Allocation

- **Proposed by Van Jacobson.**
- **Divides the address space up by session scope, and sizes the partitions according to the sessions visible in them.**



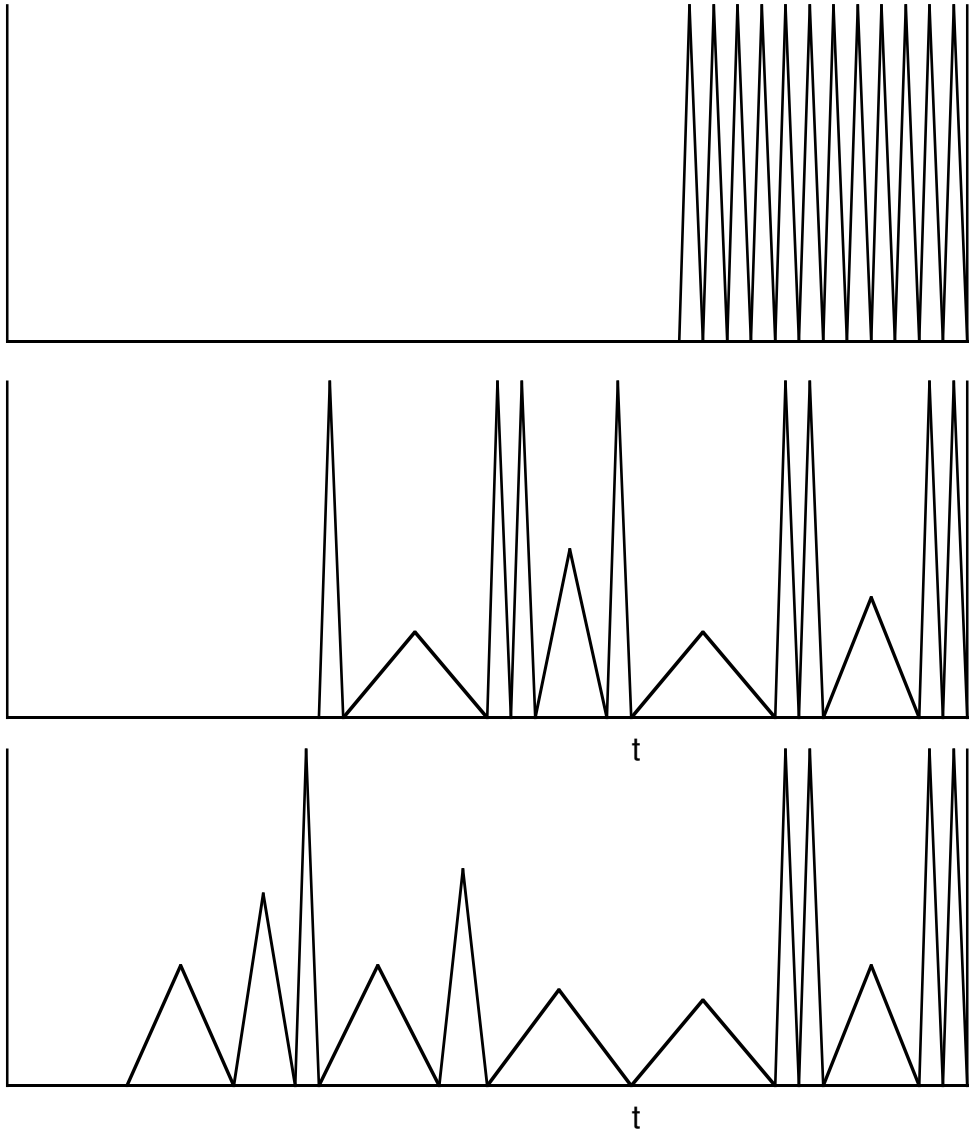
IPRMA

IPRMA still scales with close to O(sqrt(N)) because it relies on randomness for:

- **coping with scope band overlaps when session distribution at one place is different from at another**
- **coping with failures in the announcement mechanism**

The scope band overlap problem can be solved, but to work it depends on improving the announcement mechanism.

- **Initial empty scope bands are at high ttl end of range**
- **As sessions are allocated, expanding higher ttl push lower ttl bands downwards**
- **Sites that can reach each other at ttl t have the same scope bands above ttl t, so unlike IPRMA no inter-band clashes can occur.**



Architecture

- **Standalone session directory instantiations will have too great a start-up delay to be able to safely perform this allocation scheme.**
- **This is one of the motivating factors towards evolving sdr towards a client server architecture:**
  - **one or more local servers run continuously and cache incoming announcements**
  - **servers communicate by periodic announcement, which can start quickly and do exponential backoff to a base rate.**
  - **clients download session data from servers.**
  - **servers perform address allocation for clients and time average address band changes to avoid problems with temporary partitionings.**

Summary and Observations

- **ALF can and should be interpreted quite broadly.**
- **ALF applies to time as well as space.**
- **Although ALF is a design principle that can lead to tools that perform well, it's difficult to draw guidelines or make generalisations from many ALF applications.**
- **Designing collaboration tools from the network constraints upwards seems to result in network tools that perform better and therefore satisfy users better than designing them from the user requirements down.**