Session Directories and Multicast Address Allocation

Mark Handley
USC/ISI
mjh@isi.edu

A Little History

- The first multicast session directory was *sd* from Van Jacobson and Steve McCanne at LBNL in 1992.

- I wrote *sdr* in 1994, and eventually it came to replace sd.

Both tools multicast periodic messages to a well known group to announce multicast sessions. They also use the information in these announcement messages to perform multicast address allocation.

Van Jacobson briefly described an address allocation scheme for sd in his Sigcomm 1994 tutorial, and this work stems from that original idea.

Basic Address Allocation Idea

Sdr announces which multicast addresses are in use. To allocate an address, simply listen to know which addresses are used, choose randomly from those not in use, and announce the address you choose.

Problem:

- Not all sessions you might clash with can be seen and so avoided. If I allocate a global scope address, it can clash with local TTL-scoped sessions elsewhere.

TTL Scoping

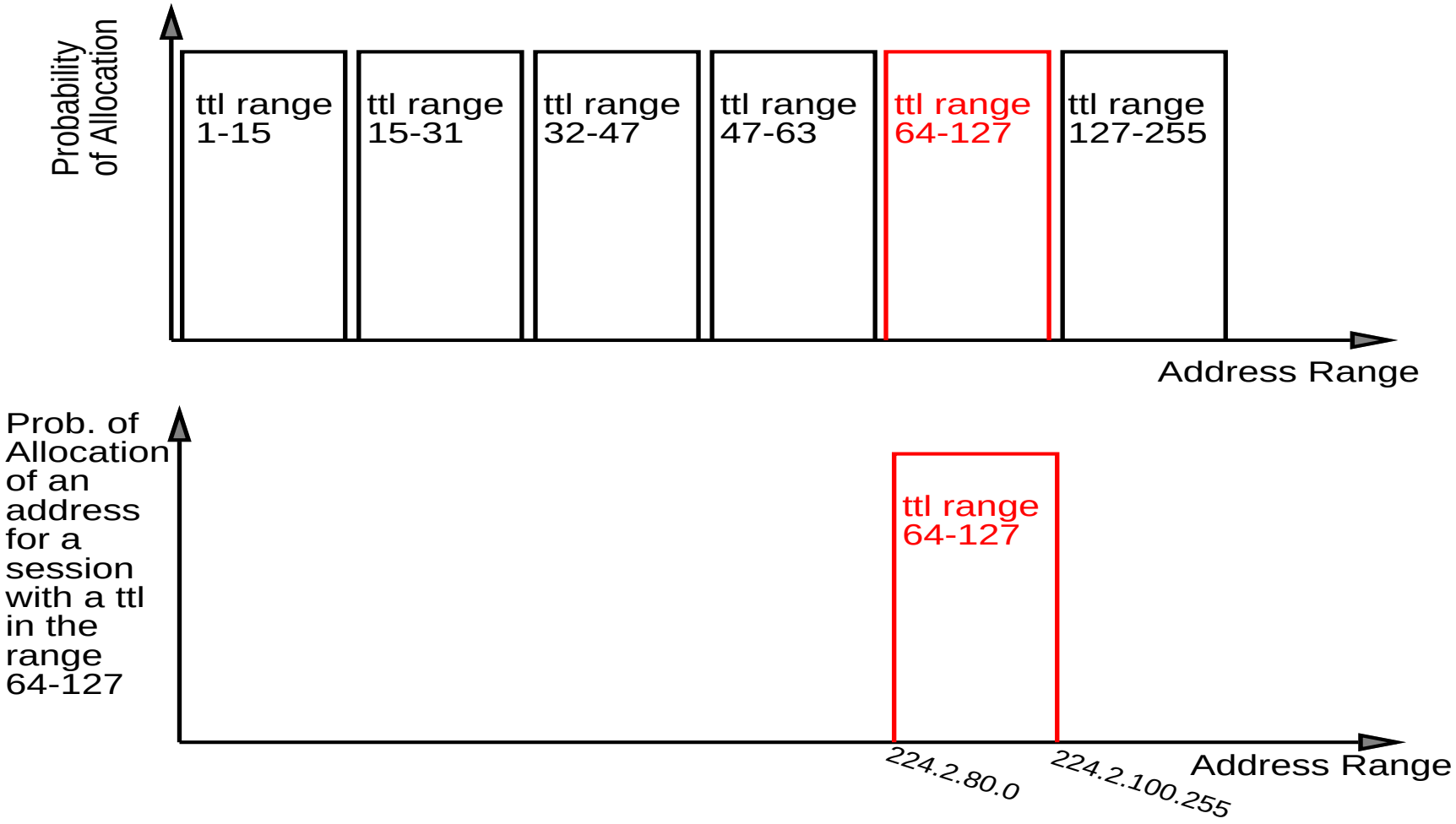
- Sender sets IP "time-to-live" field in packets.
- Each router decrements TTL.
- When TTL reaches zero, packet is dropped.
  - *TTL constrains how far a multicast packet can travel.*

Configured TTL thresholds in routers supplement the basic mechanism to divide the Mbone into appropriate scope-zones such as "Europe" or "site-local".

- When a session is created, the TTL to decide its scope is chosen.
- Announcement packets go to the same scope as the session they announce.

Informed Partitioned Random Multicast Address Allocation (IPRMA)

To avoid clashes, split the address space into different ranges for each scope:



The Problem with IPRMA

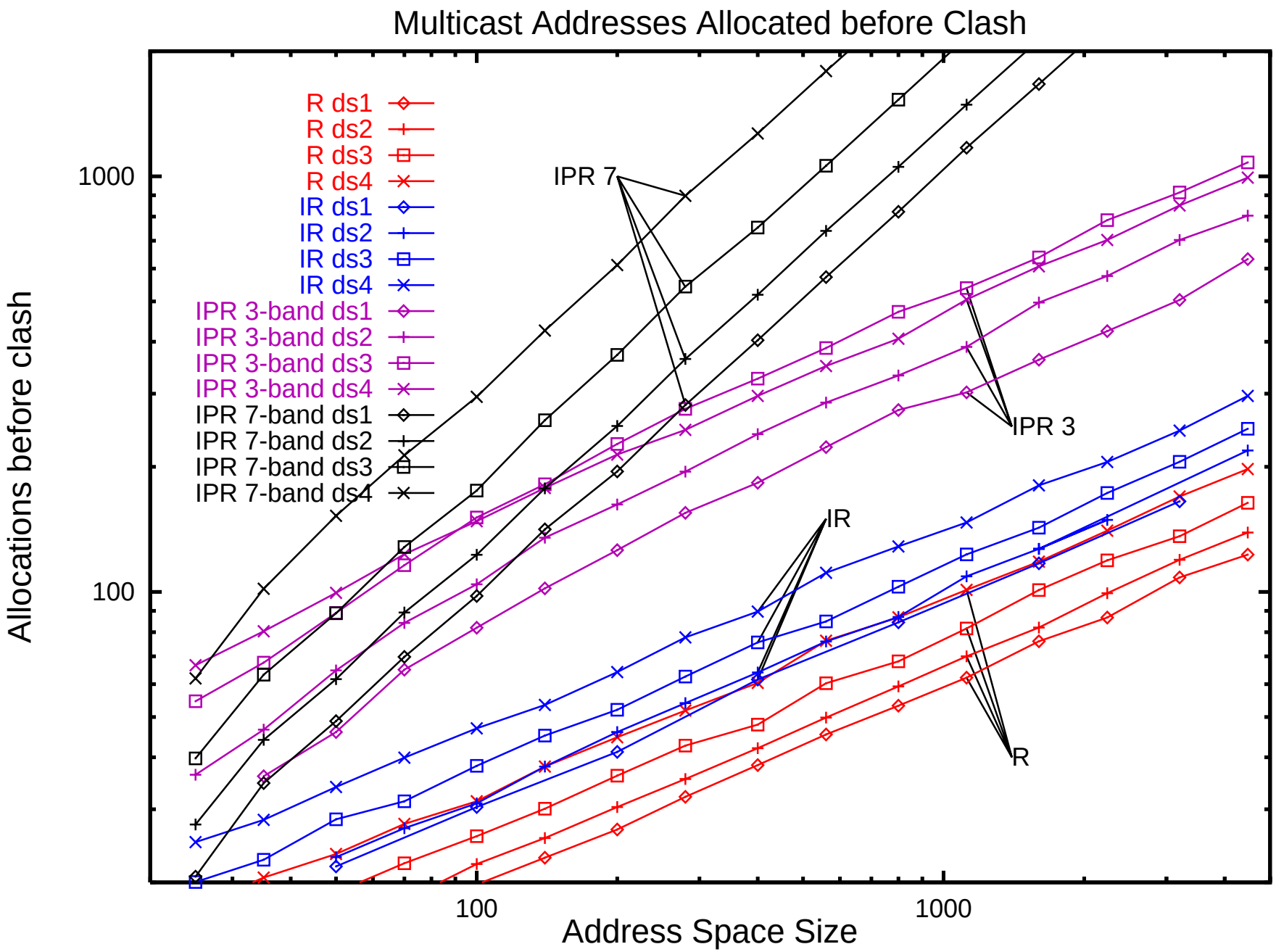
Not all locations choose the same TTL thresholds:

- Within Europe, TTL 48 divides countries.
- In North America, TTL 48 thresholds are not used.

How then do you partition the address space?

- Too many ranges result in many empty ranges and poor utilization.
- Too few ranges result in failures of the informed mechanism.

Informed Mechanism Failure



Adaptive Address Space Partitioning

If we don't know in advance the TTL values used worldwide and the distribution of sessions between them, it makes sense for the partitioning to be adaptive:

- Grow a partition when many addresses are assigned from it.
- Shrink unused partitions.

Van first proposed this, but the difficult part is doing it without having a partition at one site grow to overlap a different scope partition at another site.

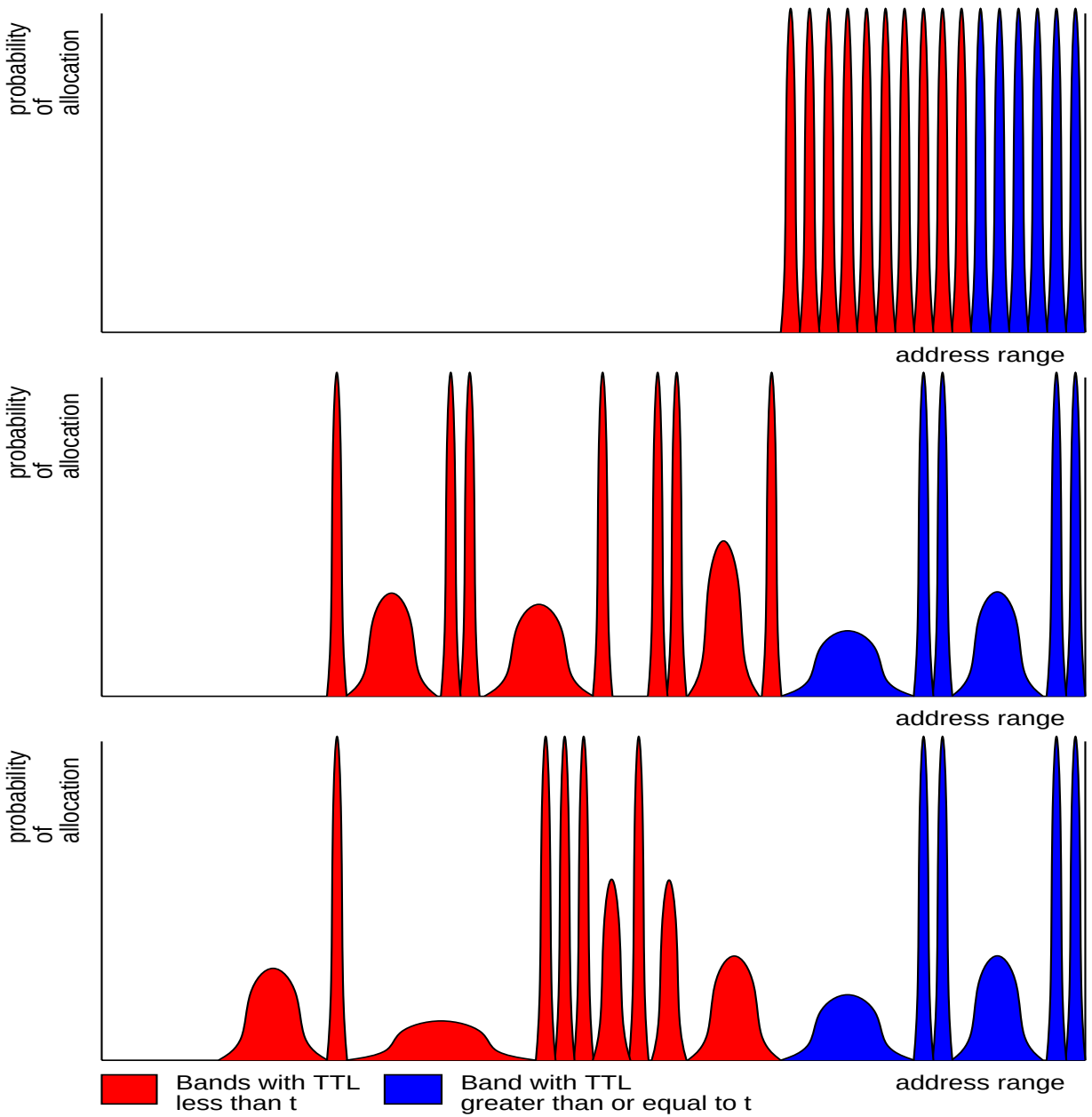
- When this happens the "informed" part of the mechanism fails.

Deterministic Adaptive Partitioning

To perform address allocation so that heterogeneity of distribution of sessions between partitions does not cause partition overlap:

- Allocate numerous partitions at the high-TTL end of the space
- Have occupied partitions "push" the lower-TTL partitions down the space.
- Have inter-partition gaps between occupied partitions to cope with short-term variations of allocation.

Deterministic Adaptive Partitioning



Simulations of Adaptive IPRMA

Adaptive IPRMA is designed to work well where the number of addresses allocated from a partition is to some extent stable. We can't simulate this by allocating until a clash occurs.

New criterion:

- *An address allocation scheme is acceptable if during the mean lifetime of a session, the probability of an address clash anywhere in the world is less than 50%.*

Simulations

Rough Algorithm

- 1. Allocate *n* sessions; reallocate until no clashes.
- 2. Delete one session.
- 3. Add one session.
- 4. Repeat from 2 until *n* sessions have been replaced.

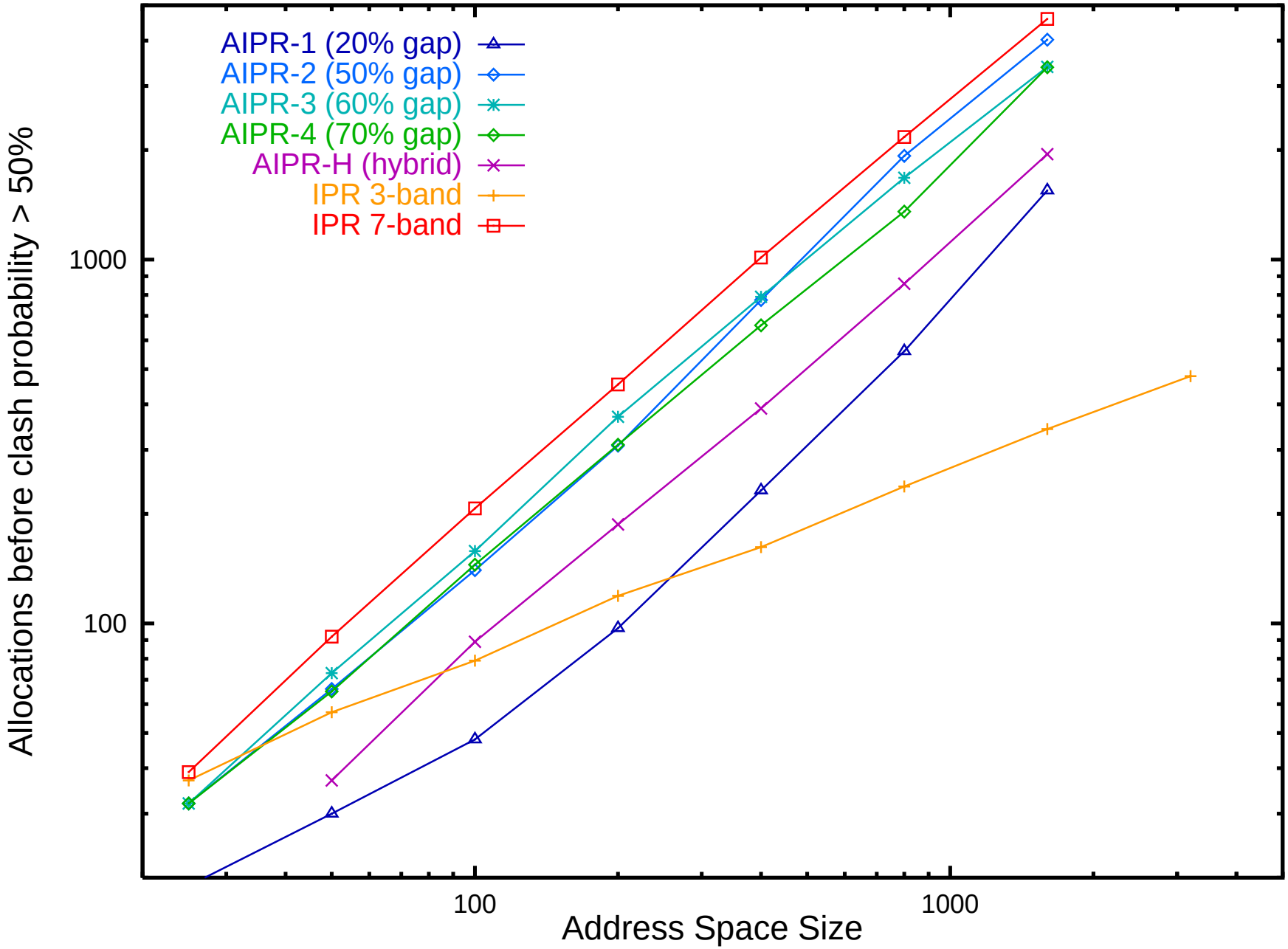
Simulations without communities:

- New session is unrelated to old session.
  - No stability in number of "local" groups

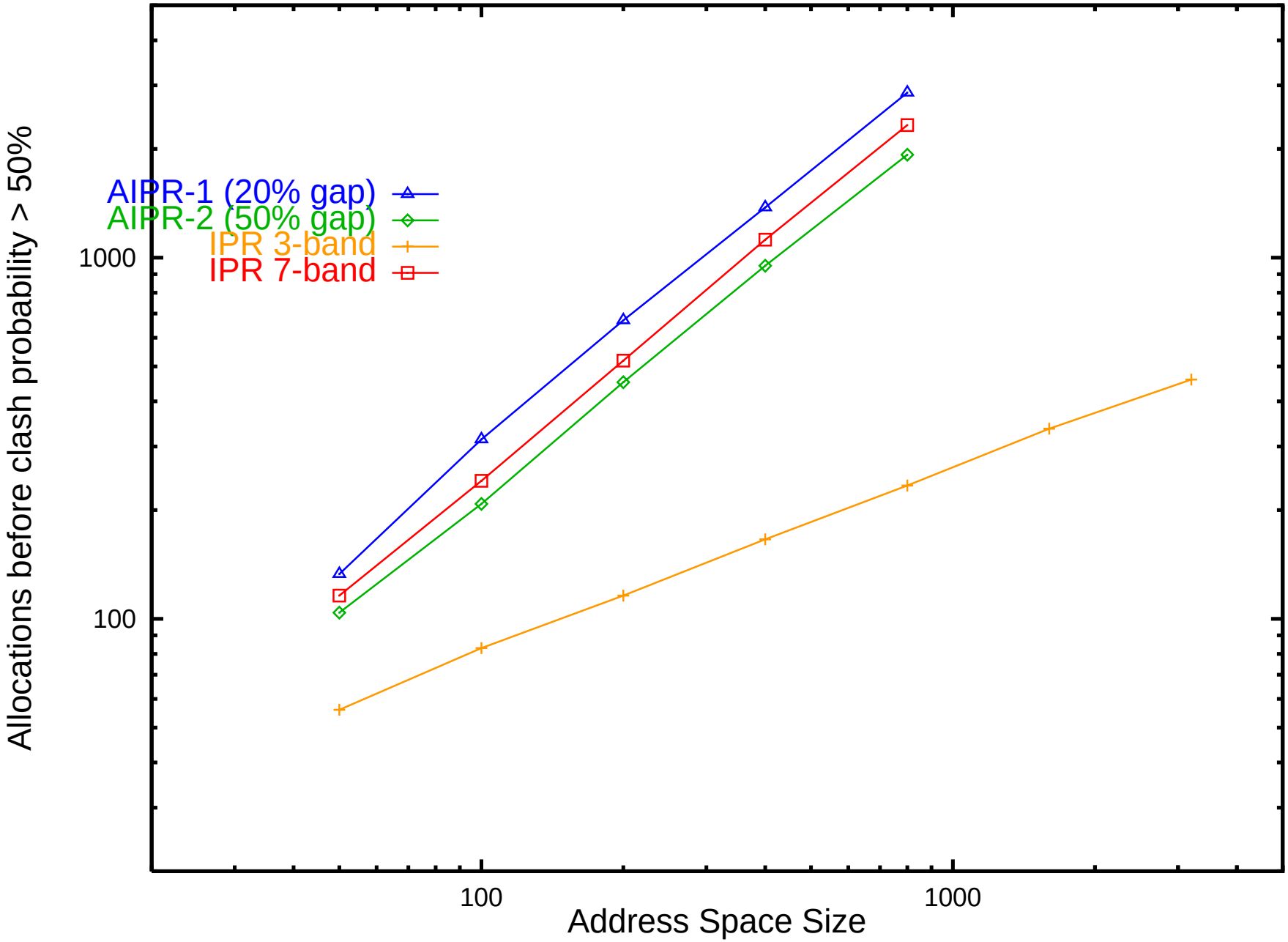
Simulations with totally stable communities:

- New session has same TTL and originator as old session.
  - Group distribution seen from each site fixed.

Simulations: No local communities



Simulations: Stable communities



Reality?

How can we simulate the locality of future use?

- *We can't we could only speculate.*

The two simulation sets give lower and upper bounds for performance.

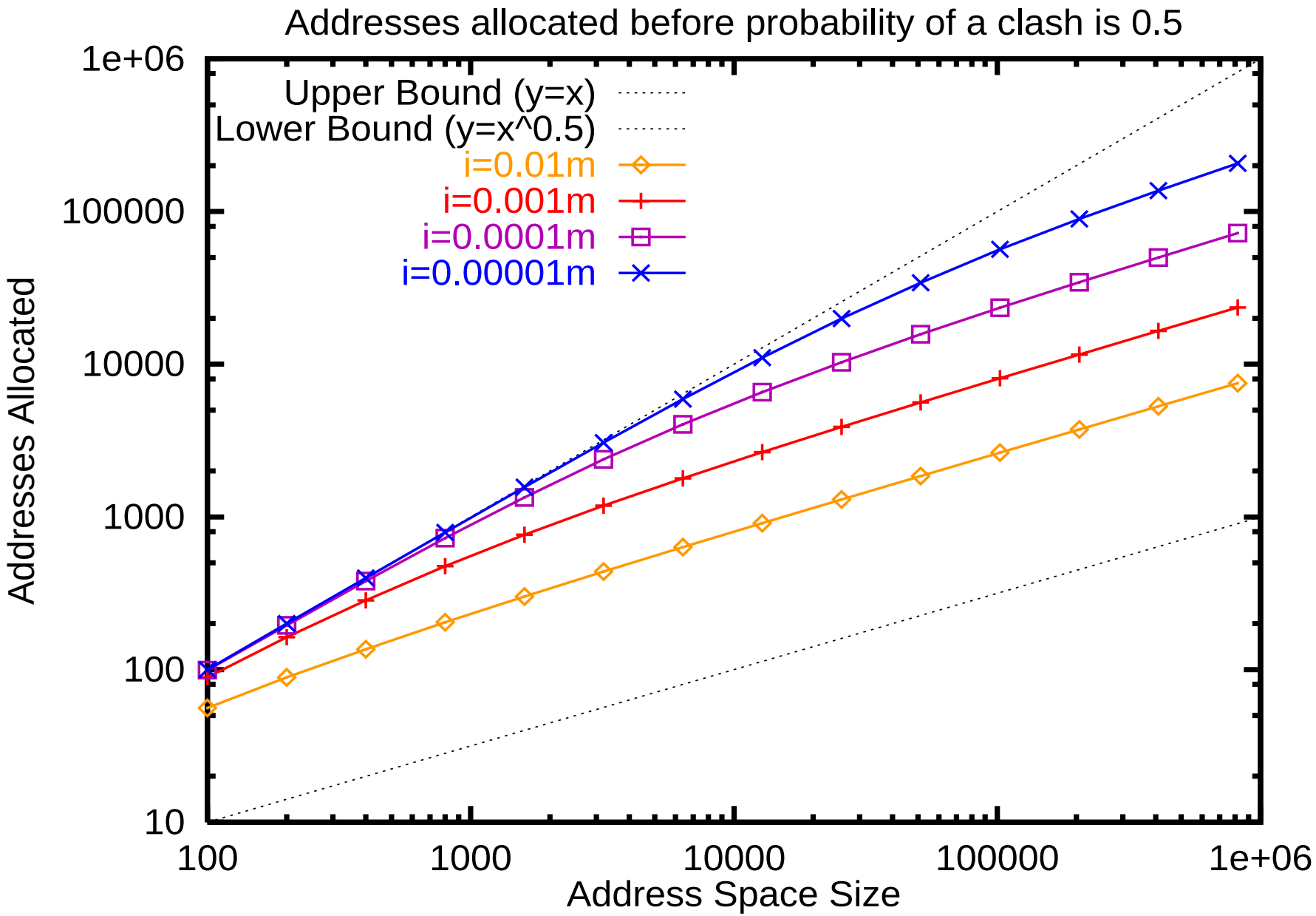
- Both scale ***O(n).***
- Static allocation slightly outperforms adaptive allocation if we know the number of active partitions and their boundaries and their are no communities.
  - In practice we don't know this.

Thus the adaptive algorithm works well given the constraints.

Packet Loss and Propagation Delay

- The previous simulation ignores propagation delay and packet loss.
- Assume:
  - mean session length: 2 hours
  - mean advance announcement: 2 hours
  - mean end-to-end delay: 200ms
  - mean packet loss rate: 2%
  - each announcement resent every 10 mins
- Mean end-to-end delay ~(0.98*0.2)+(0.02*600) = 12 secs
- Therefore approximately 0.1% (12 secs/4 hours) of sessions are not visible at any time.

Packet Loss and Propagation Delay



Packet Loss

- Instead of periodically announcing a session with a fixed base rate, start with a high rate (e.g. 1 per second) and exponentially back off down to a fixed base rate.

- With global propagation delay, this brings us to approximately $i=0.0005m$
  - Scales well to around 10000 addresses allocated in a partition.

- ***This is not good enough!***

Two Solutions

- Introduce *hierarchy* to address allocation.
  - Do distributed address-range allocation between domains on a long timescales to mitigate propagation delays and loss.
  - Do distributed individual address allocation within a domain on short timescales from the address-ranges.

- Add scalable mechanisms to *detect and correct* clashes when they do occur.

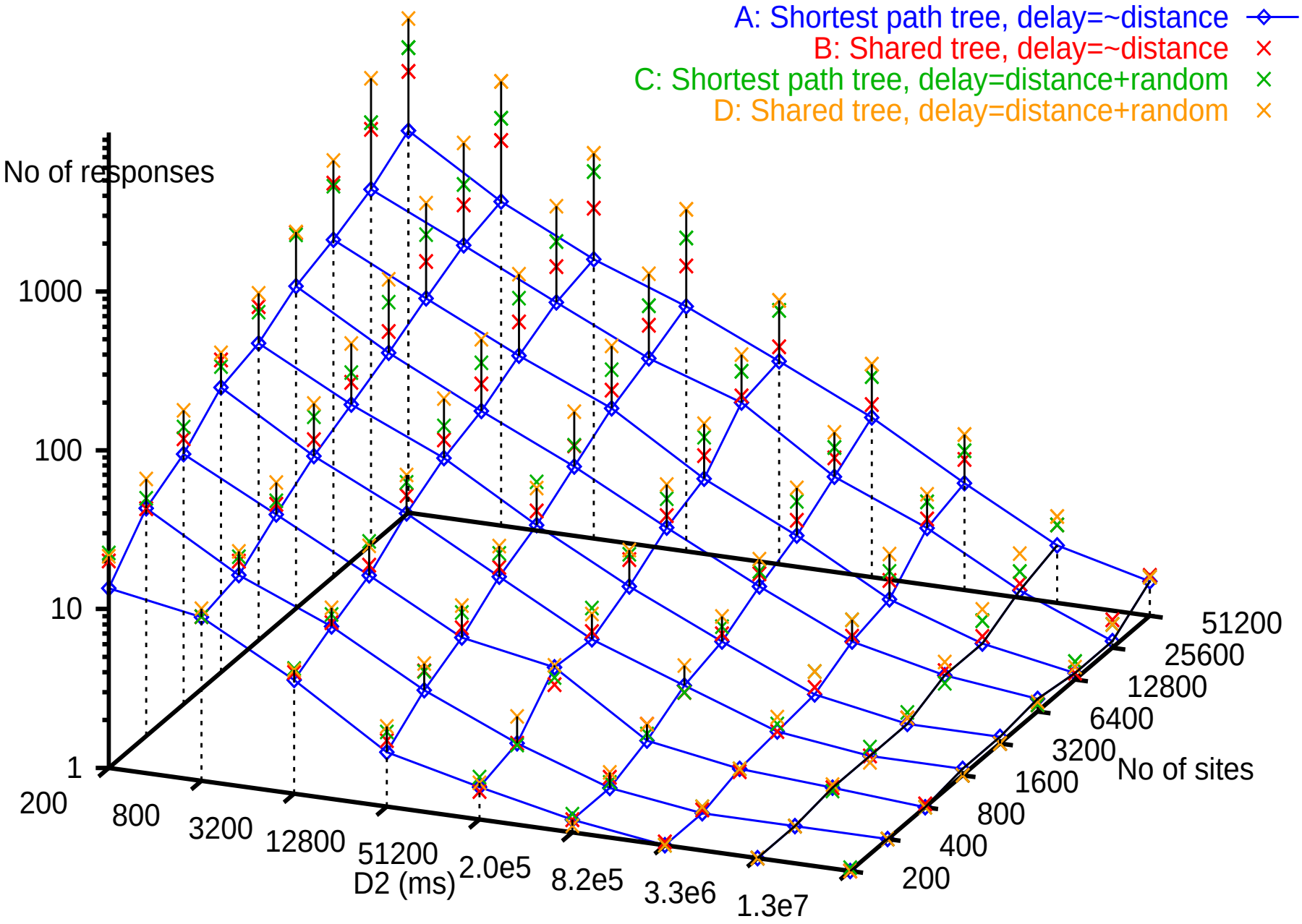
Detecting Clashes

- We'd like any address allocator that notices a clash to be able to report it if the originator doesn't.
  - There may be millions of allocators.
  - ***How do we avoid clash report implosion?***
- SRM (Sigcomm '94) uses random timers chosen uniformly from a range [*d1*:*d2*].
  - *d1* and *d2* are chosen based on measurements of propagation delay.
  - One multicast response suppresses others.
- ***What can we do with no estimate of the population and no delay matrix?***

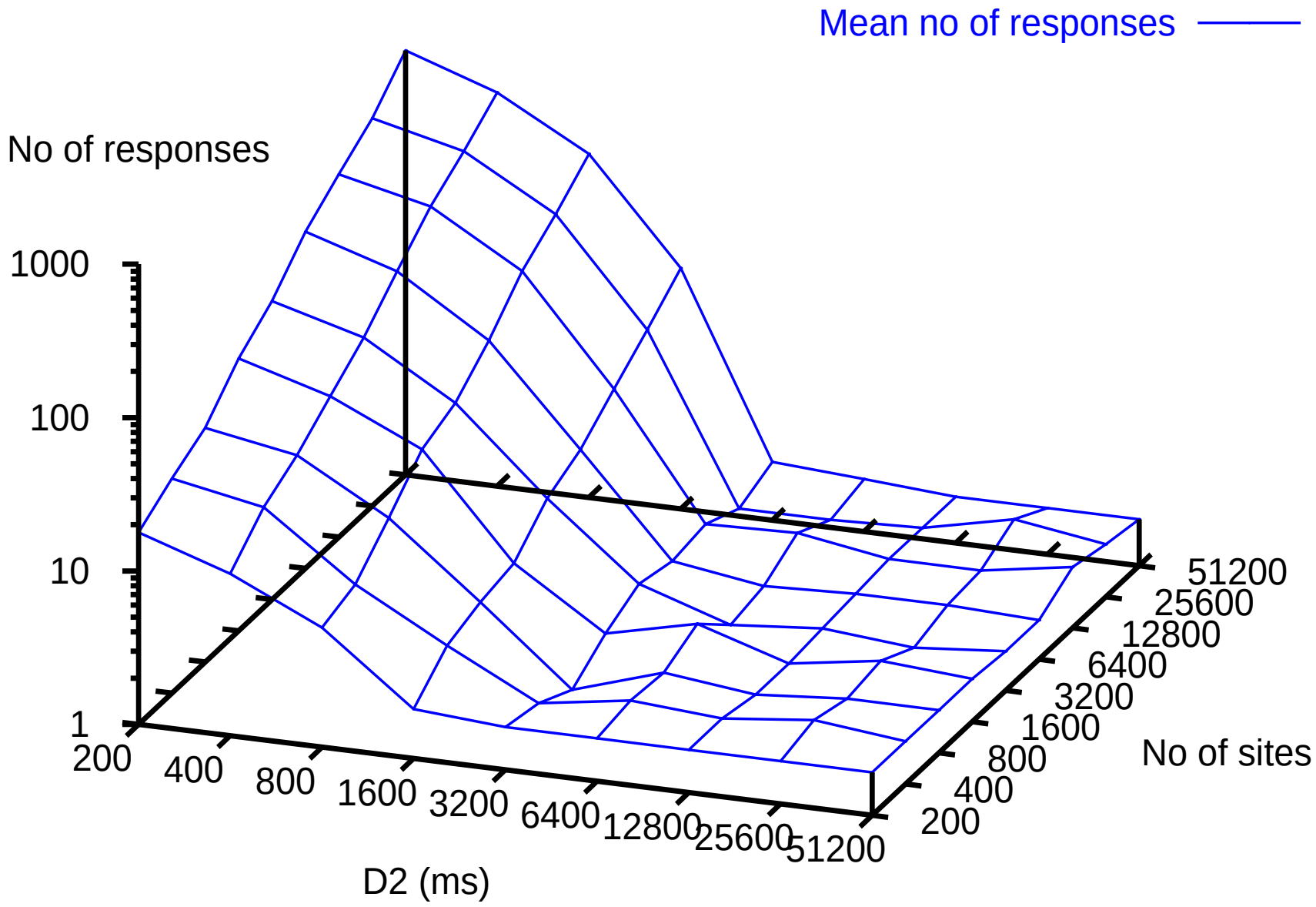
Exponential Random Timers

- Uniform random timers are too dependent on knowing the number of nodes trying to respond.
- If we use exponentially distributed random timers we can largely remove this dependence.

Simulation: Uniform Random Timers



Simulation: Exponential Random Timers



Summary

- We’ve designed a scalable approach for adaptive partitioning of the address space to deal with allocating addresses to TTL scoped sessions.
  - The real problem is propagation delay.

- Solutions:
  - Increase speed of light.
  - Use a hierarchy with the upper levels operating on different timescales.
  - Move away from TTL scoping.

- Within a lower level allocation domain:
  - Use exponential backoff between announcements
  - Use exponentially distributed random timers to send clash detection messages.