Rumor Riding: Anonymizing Unstructured Peer-to-Peer Systems

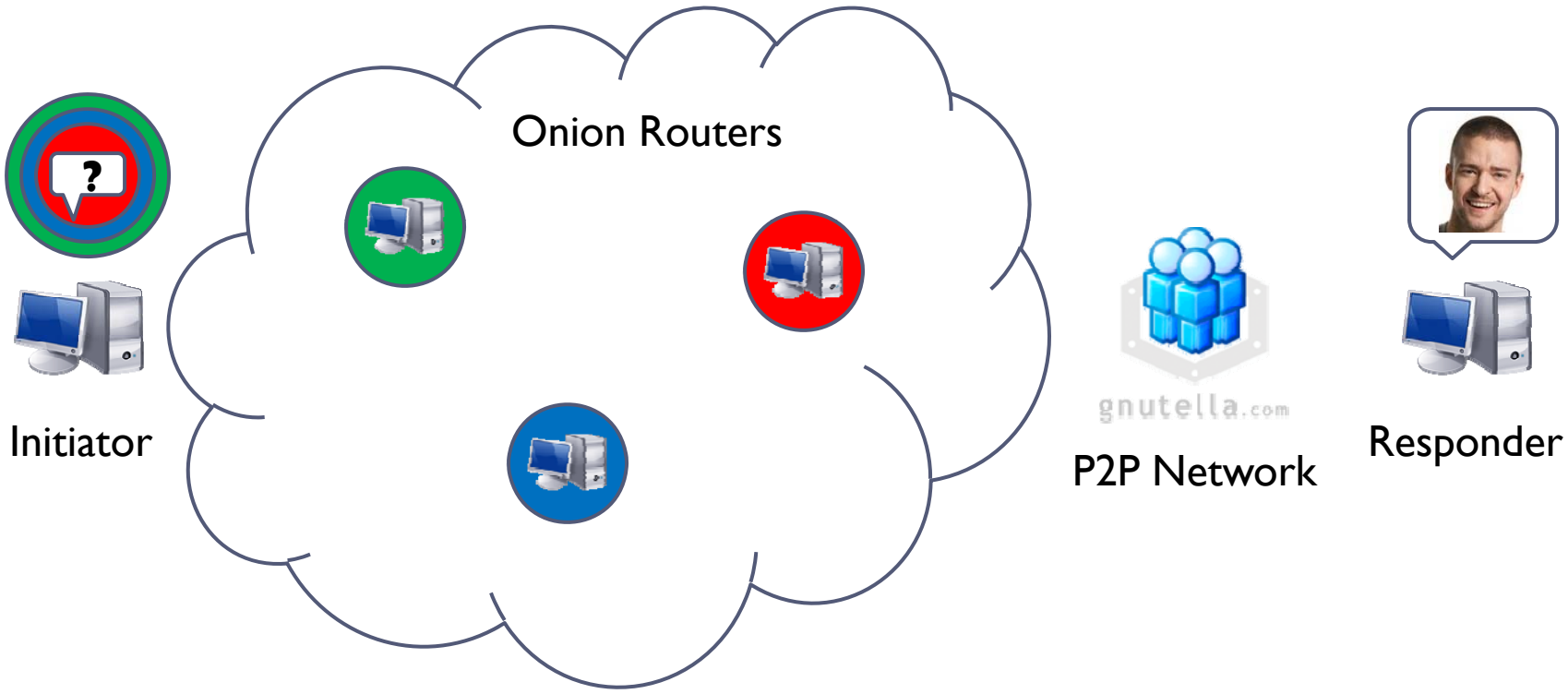
Narrated by Christo Wilson

Table of Contents

Problem Scenario

- You want a copy of that new Justin Timberlake song
  - Too embarrassed to get it from a store
  - RIAA fueled by devouring human souls
- What is needed: Anonymization
  - Existing protocols are unsuitable for P2P
    - Path-based
    - Heavyweight, asymmetric layered encryption
- Proposed solution: Rumor Riding
  - Non-path based (sort of)
  - Uses Symmetric encryption (mostly)

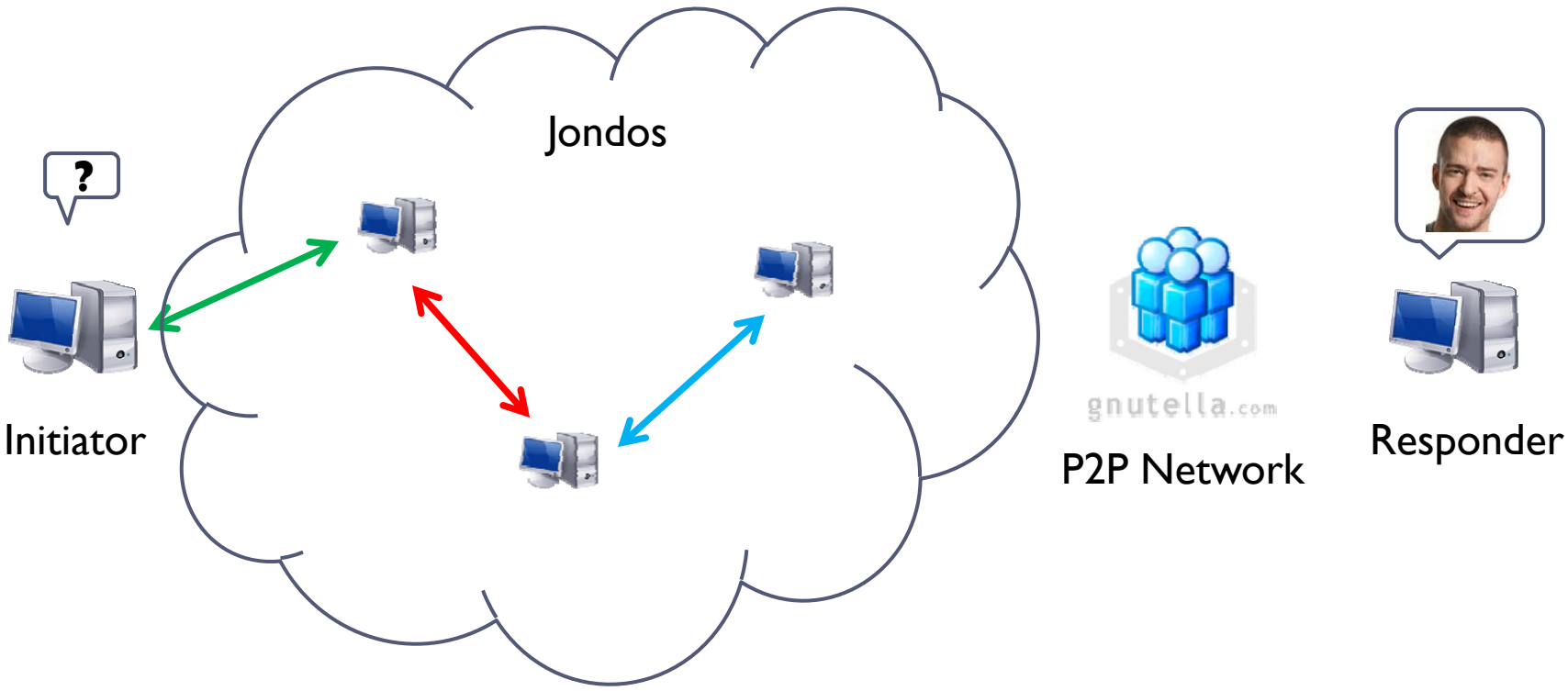
Existing Schemes: Tor



Existing Schemes: 

- Why not use Tor for P2P?
  - Designed for client-server architectures
    - No responder anonymity by default
    - Could be re-architected to fix this deficiency
  - Asymmetric decryption at every hop
  - Key exchange nightmare
  - Pathing:
    - Construction requires knowledge of many peers
    - Must be persistent for duration of file transfer
    - Paths must be explicitly rebuilt periodically to maintain anonymity

Existing Schemes: Crowds



Existing Schemes: Crowds

- Why not use Crowds for P2P?
  - As with Tor, designed for client-server architectures
    - No responder anonymity by default
    - Could be re-architected to fix this deficiency
  - Symmetric decryption at every hop provides weaker anonymity
  - Still have a key exchange nightmare
  - Pathing:
    - Must be persistent for duration of file transfer
    - Lack of source-routing provides weaker anonymity

Rumor Riding: Design Goals

- Provide high degree of initiator and responder anonymity
- Use symmetric encryption
  - Do not require extensive key exchanges
- Design with attributes of P2P topology in mind:
  - Do not require any explicit path construction
  - Require as little path-persistence as possible

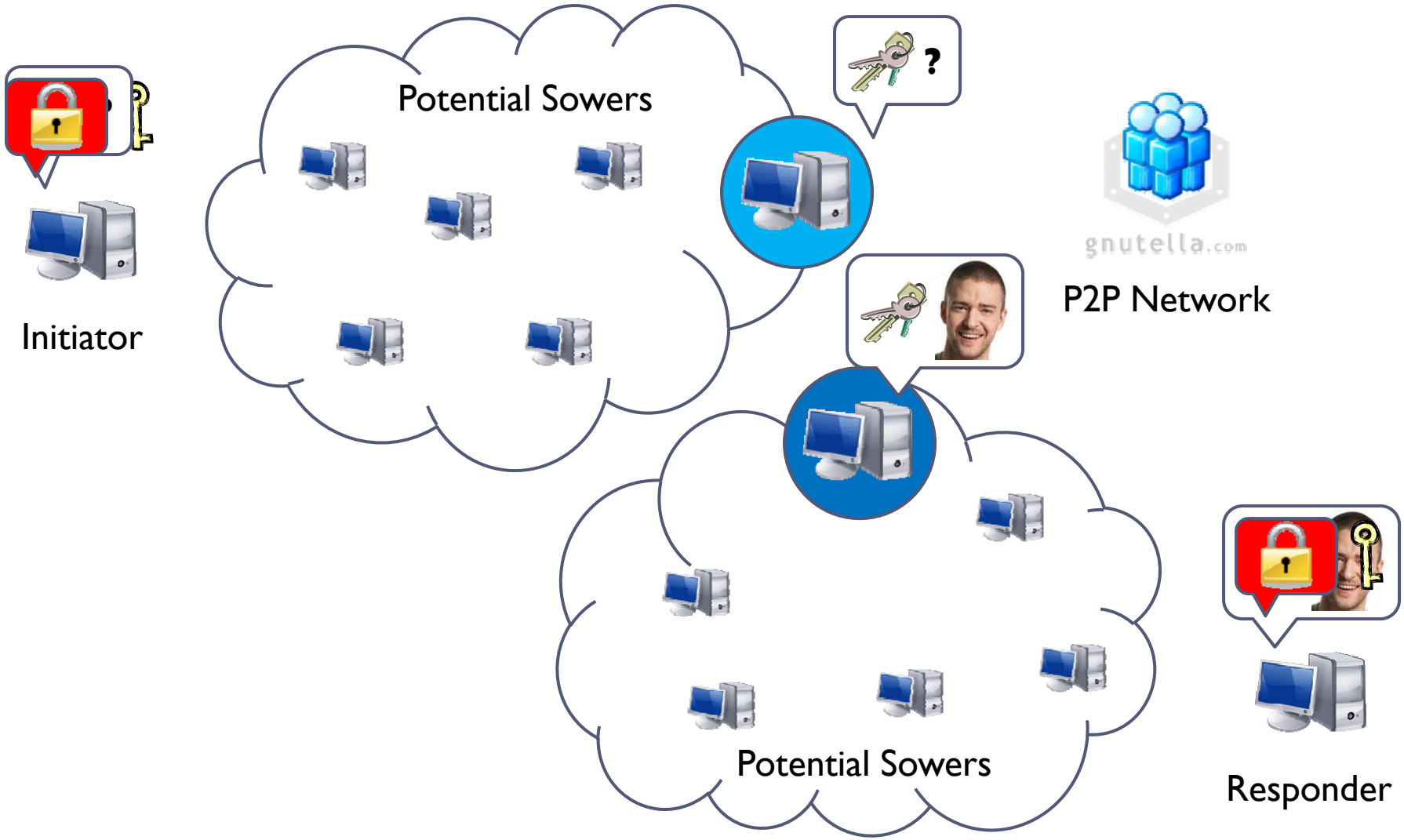
Rumor Riding: Protocol Design

- Every message split into two pieces: encrypted data and key
  - Symmetric encryption (AES, 128-bit)
  - Each piece called a rumor
- Data and key each forwarded to different neighbors
  - Rumors continue travelling outward in a random walk
- Nodes maintain rumor caches
  - Rumors constantly checked for pairings (collisions)
  - Collisions identified using CRC check
- Nodes which identify rumor collisions become sowers
  - Act as the proxy for the initiator

Rumor Riding: Protocol Design

- Conversations encrypted with public keys
  - Initial query and response include initiator and responders keys
  - 1024-bit RSA prevents eavesdropping on conversations
- Rumor convergence is controlled
  - Rumors can be issued in multiples
  - Each rumor has an adjustable TTL

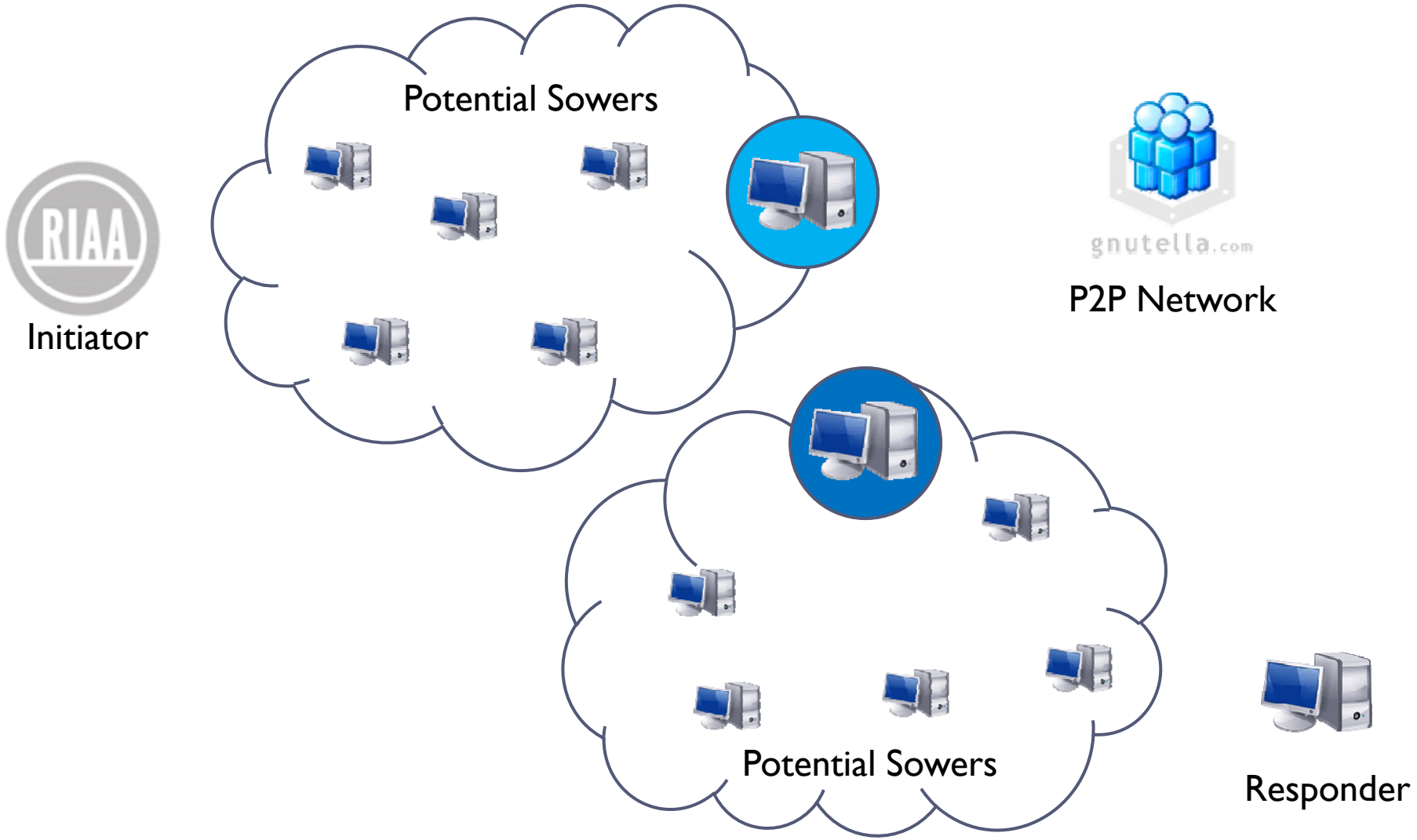
Rumor Riding: Example



Rumor Riding: Analysis

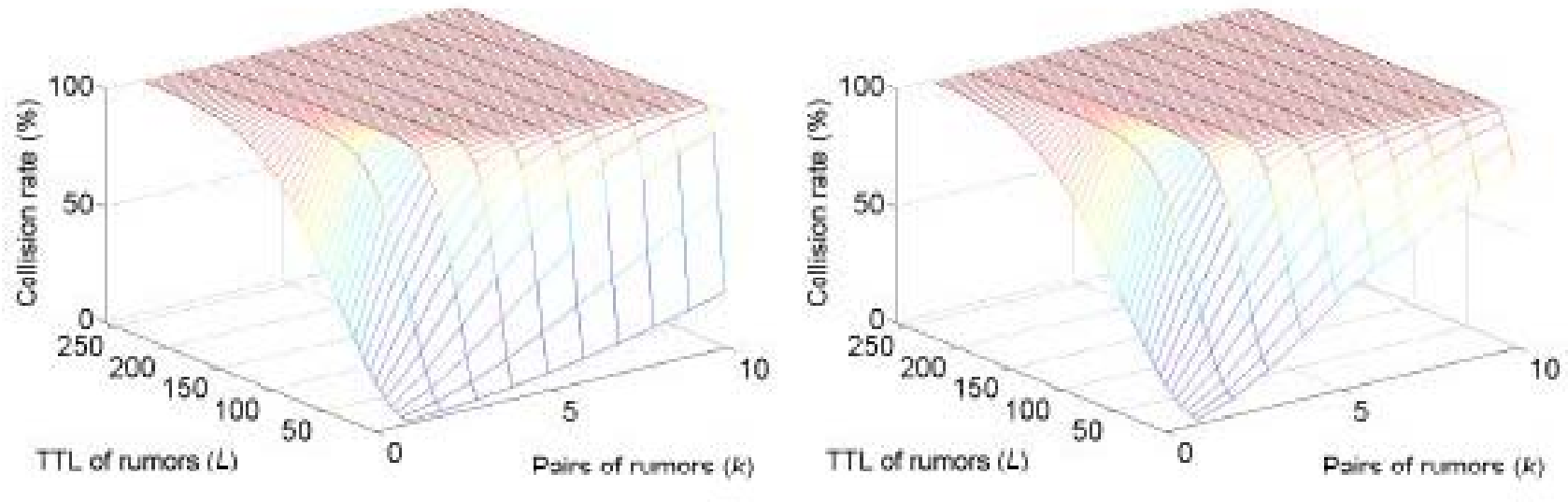
- Resilient to attack
  - Forwarding provides Crowds-like plausible deniability
  - Separating paired rumors makes local eavesdropping difficult
  - End-to-end public key encryption prevents man-in-the-middle attacks
  - Random walks prevent timing attacks and traffic analysis

Rumor Riding: Analysis



Rumor Riding: Analysis

- Trace driven simulation
  - 1,000 to 100,000 node Gnutella-like network
  - 600 second mean node lifetime
- Theoretical vs. Simulated rumor collision rates:

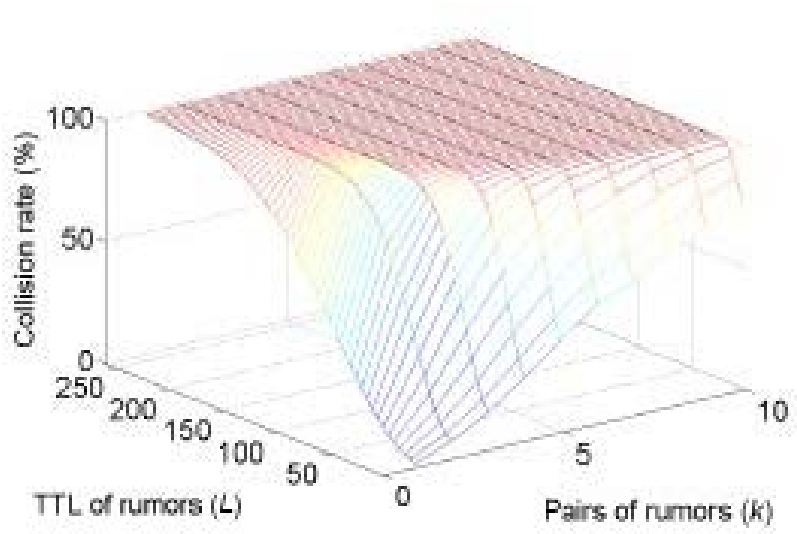


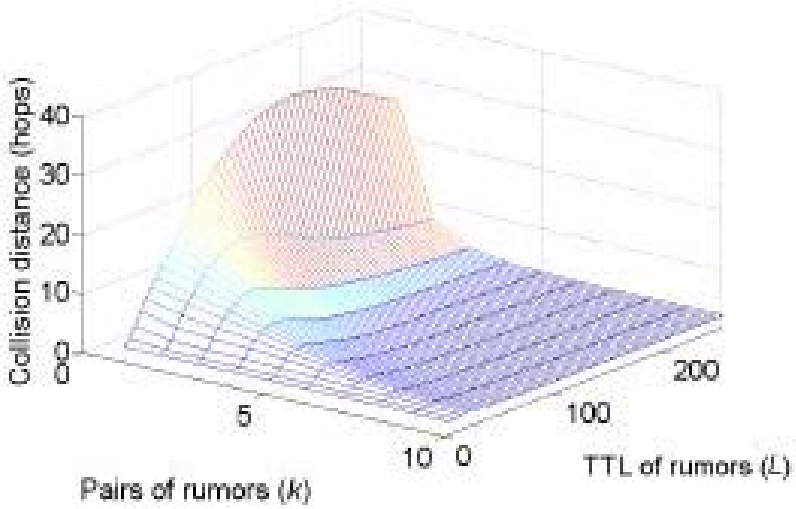
Practical Considerations

- O(n) processing overhead
  - Every incoming rumor must be decrypted and CRCed against entire cache contents
- Static RSA key pairs enables correlative attacks
  - Compromised initiators and/or responders can track remote hosts individually, uniquely
- Duplication of effort, non-unique search query results
  - Queries are usually controlled floods
  - K-Rumors can result in K sowers issuing queries
  - Each query may elicit identical responses
- File chunking necessitates return path persistence or constant production of new rumors
  - Payload rumors in multiple may result in duplicates at receiver

Practical Considerations

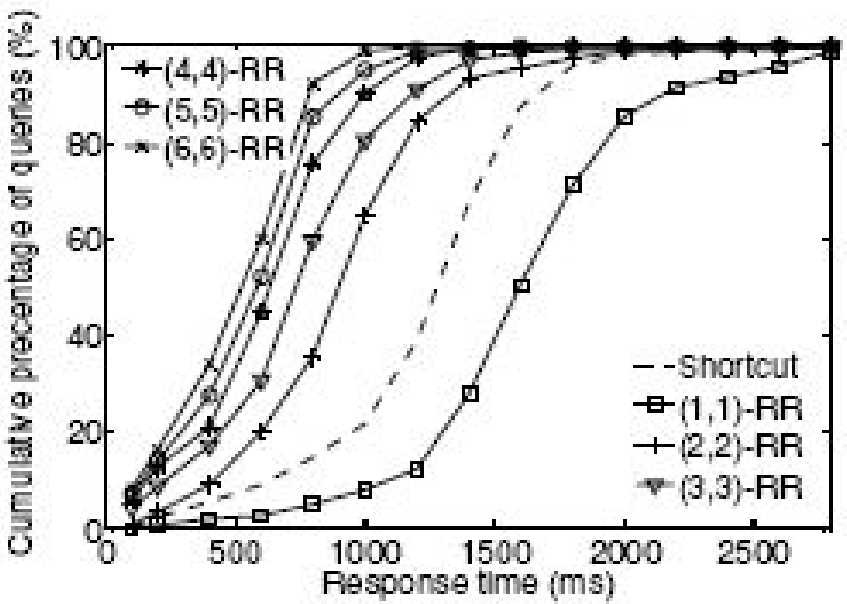
- Small-world networks significantly compromise anonymity
  - Compromised super-nodes can potentially allow statistical ascertain of initiators/responders
  - Rumor collision distance inversely related to collision rate





Practical Considerations

- Latency
  - Numbers are way higher than cumulative latencies for path-based protocols
  - This applies to file transfers too, not just queries!



Conclusions

- Novel protocol design
  - Surprising that any random walk based protocol even works
  - Decent anonymity
  - Integrates well with P2P network topologies
- Trace driven simulations help prove feasibility
- Promises of low overhead and no-pathing are overblown
- High latency and rumor generation overhead may hinder large file transfers
- Seems geared toward Gnutella-like P2P protocols
  - Would be more useful/applicable if it worked for Torrents

Questions?

- No, I don't have any Justin Timberlake for you.