Lessons Learned from Designing Byzantine-Resilient Protocols

Cristina Nita-Rotaru

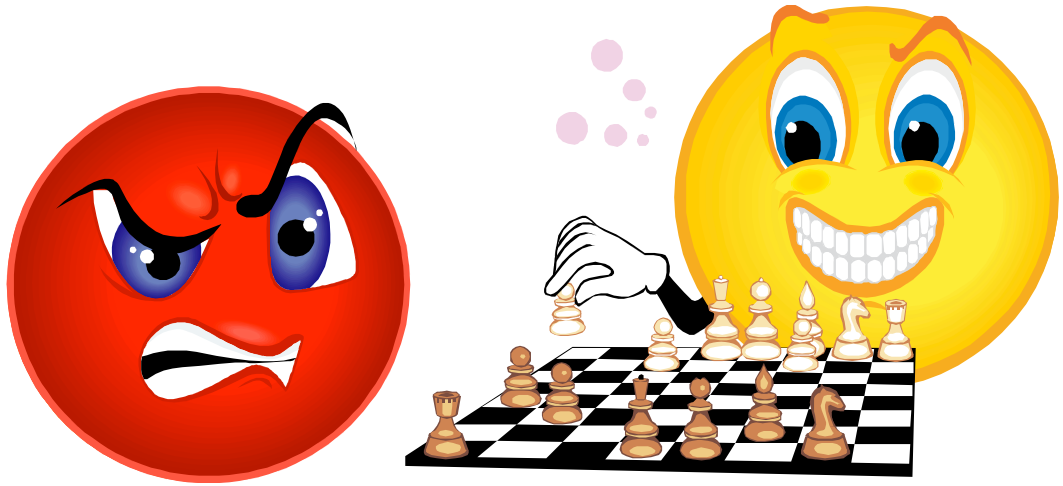
Department of Computer Science and CERIAS
Purdue University

PURDUE UNIVERSITY

CERIAS

Byzantine-Resilient Protocols

- Assume some of the protocol participants (insiders) do not play by the rules
- Provide correct service to correct participants in spite of inside adversaries

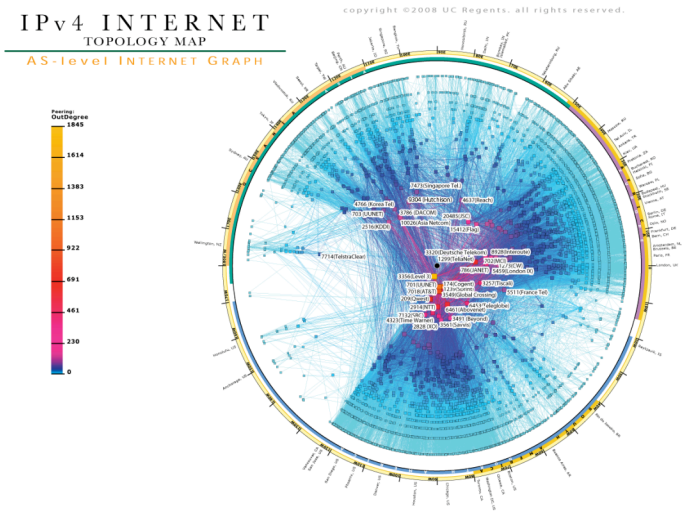


Once Upon a Time: Byzantine Generals Problem

- Agreement in the presence of arbitrary faults [LRP82]
  - Correct processes reach agreement
  - Malicious processes lie, do not send messages, show two-face behavior
- Require two thirds of processes to be correct

Byzantine-Resilient Routing

- Robustness of routing in the presence of malicious routers: focus on data delivery
- Assumes link-state routing [Per89]
- Uses flooding and agreement between neighbors to guarantee delivery



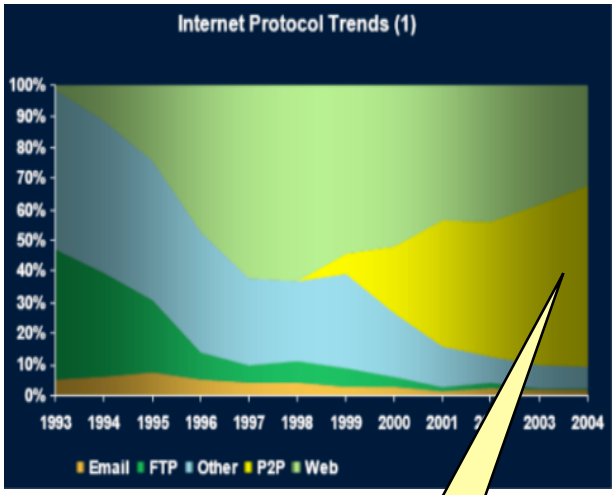
www.caida.org

The Byzantine General of Today

- Exploits software vulnerabilities to compromise a computer (these are not arbitrary faults anymore)
- Plays subtle games: timeouts, adaptation, stealth attacks

Target Applications

- Replication services
- Routing and data delivery
- P2P applications (video, file sharing, VoIP)
  - Structured overlays
  - Unstructured peer-to-peer systems continuously under attack



P2P traffic

Challenges in Designing Byzantine-Resilient Protocols

- Limited scalability
  - High communication cost:
  - High cost regardless of the presence of adversaries in the network
- Difficult to delimitate correct behavior from an incorrect one
  - Many nodes collude
  - Not enough history is available
  - Single source of information

This Talk …

- Share what we learned from our work on Byzantine-resilient protocols, using two scenarios:
  - Byzantine-replication services
  - Unstructured peer-to-peer multicast overlays

Scalable Byzantine-Replication Services

Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage

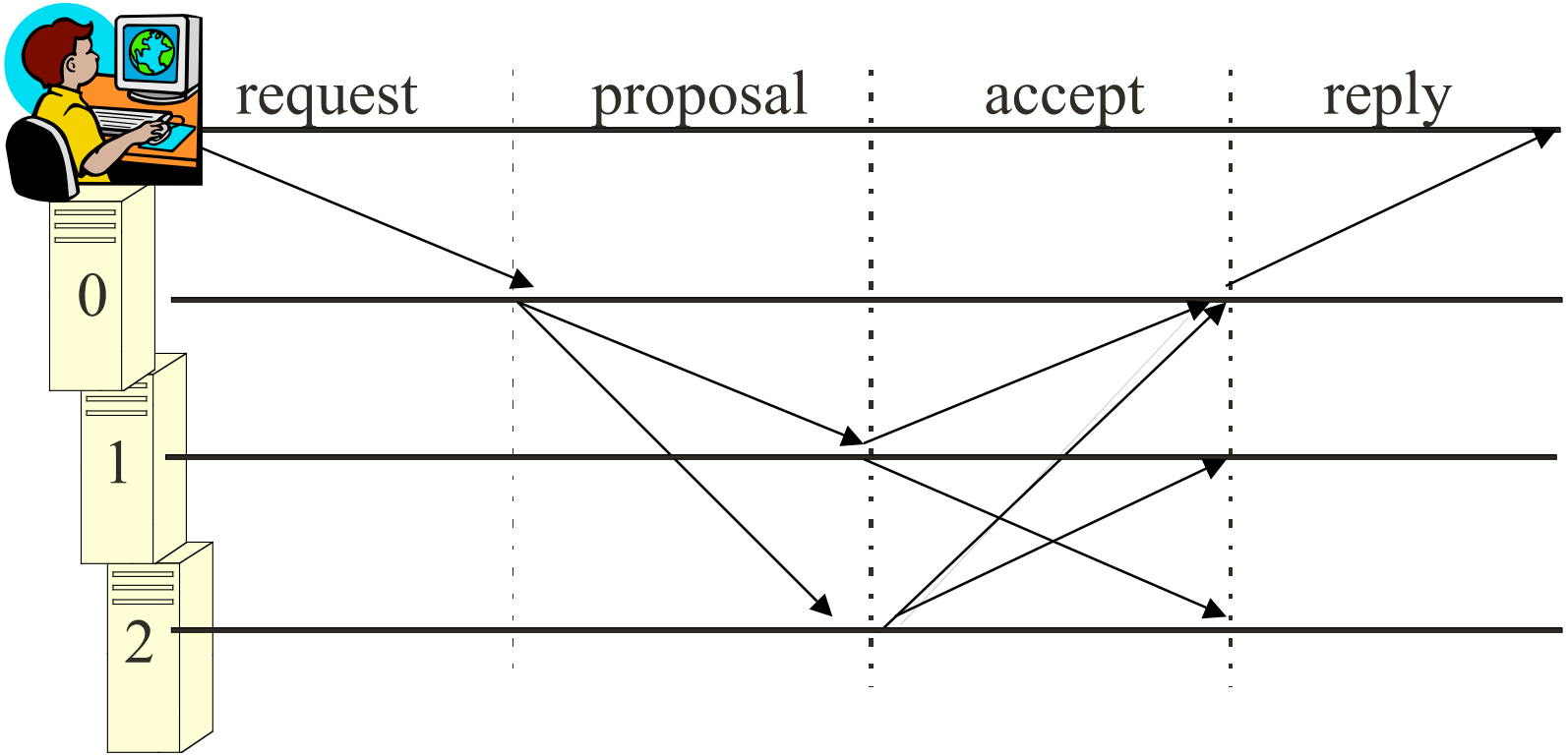
State Machine Replication [Sch90]

To tolerate f faulty servers

Benign faults: Paxos [Lam98,Lam01]:
must contact f+1 out of 2f+1 servers and uses 2 rounds to allow consistent progress, 1 answer needed by a client

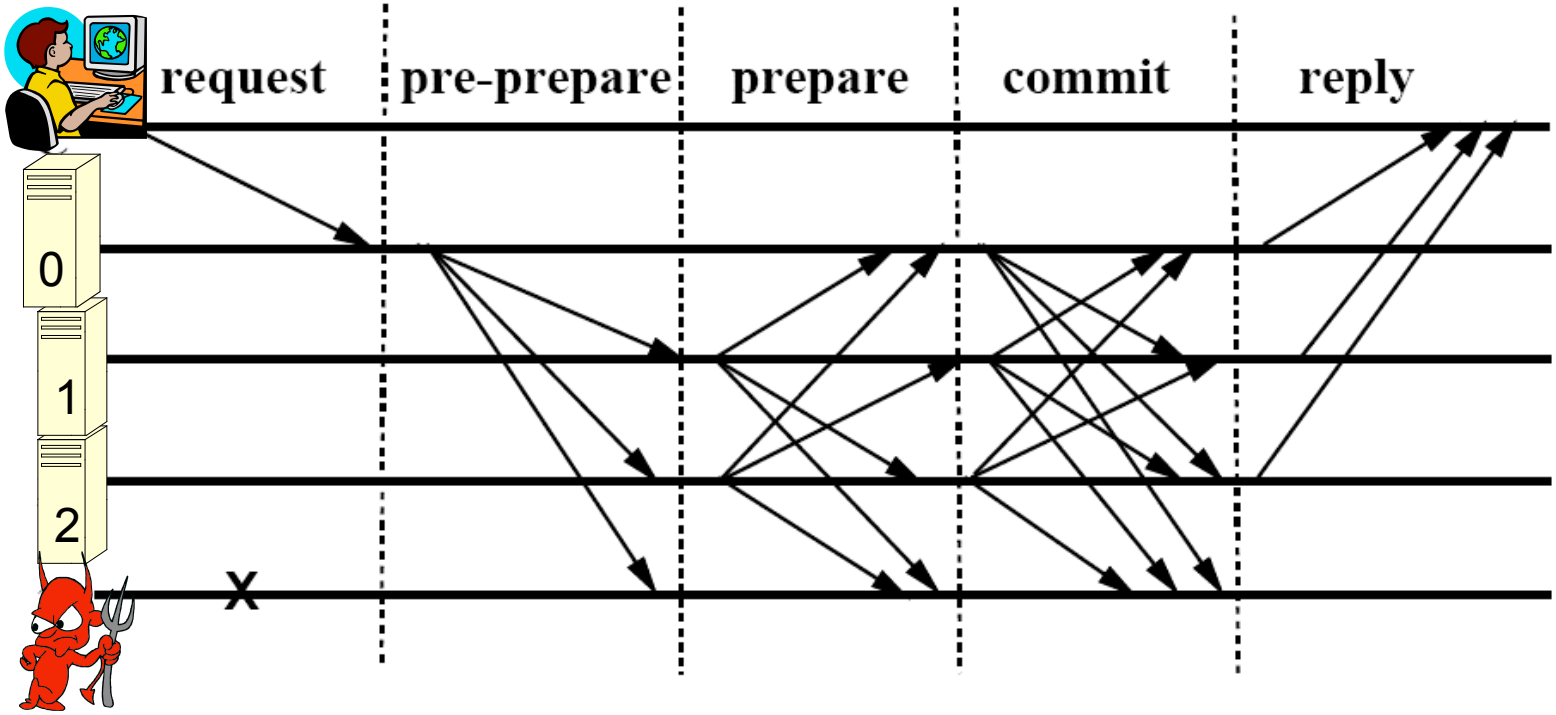
Byzantine faults: BFT [CL99]:
must contact 2f+1 out of 3f+1 servers and uses 3 rounds to allow consistent progress, f+1 identical answers needed by a client

State-of-the-Art: Paxos [Lam98]



f servers can crash, f=1 in this example

State-of-the-Art: BFT [CL99]



f servers are malicious, f=1 in this example

Limitations of Current Solutions

- Li
-
  - **Zi** **progress** and f+1 for the client to obtain a correct answer
  - On WANs: Partitions are a real issue, clients depend on remote information, long delays

FLAT ARCHITECTURES
DO NOT SCALE

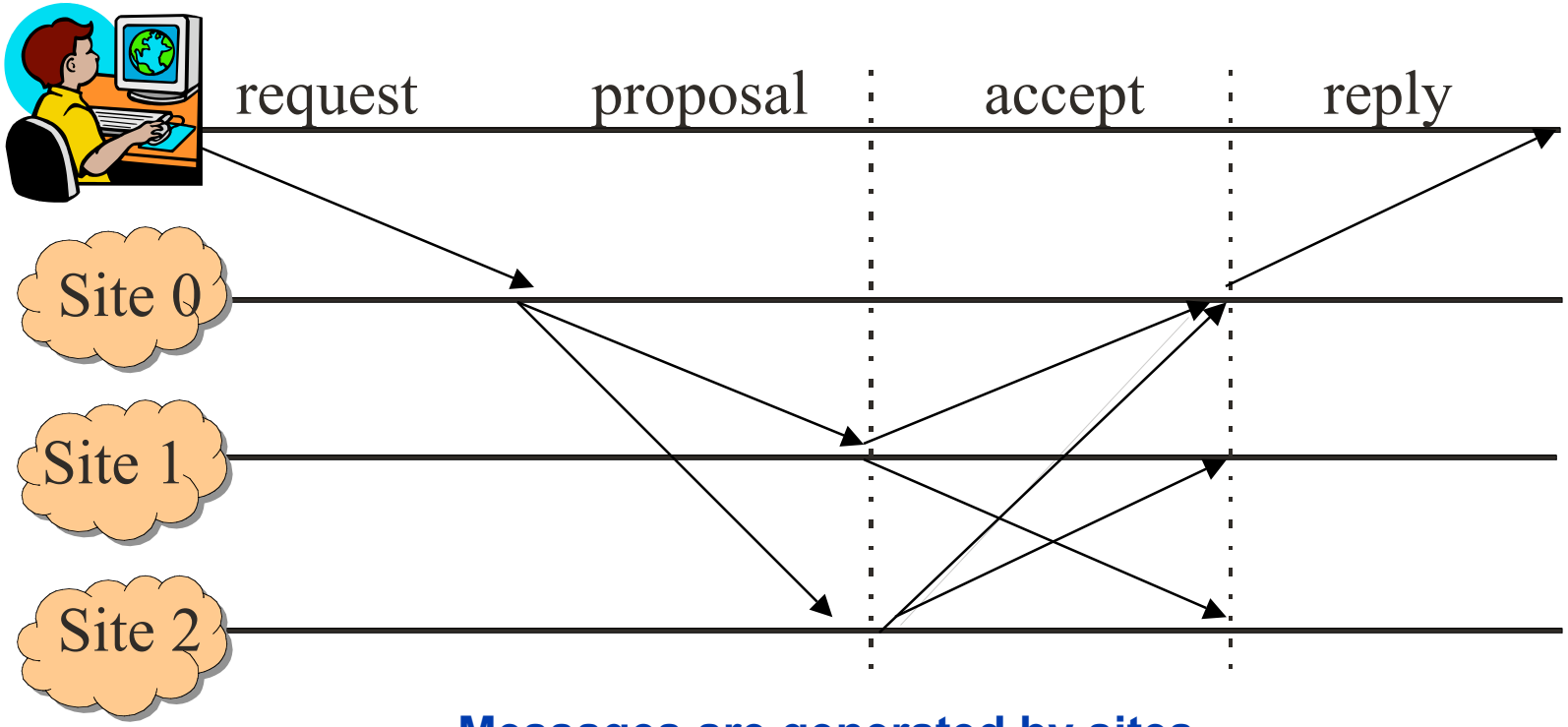
Use an Hierarchical Architecture

- **Introducing Steward:**
  - **Uses a hierarchical architecture**
  - Global protocol between sites (masks benign faults)
  - Local protocol within a site (masks Byzantine faults)
  - **Result**: less messages and one communication round less in wide area networks
- Requires more hardware

Advantages of Steward Architecture

- **Reduces the message complexity** on WANs exchanges from $O(N^2)$ to $O(S^2)$
- **Improves the availability** of the system over WANs: f+1 of connected sites needed to make progress, instead of at least 2f+1 servers (out of 3f+1) in flat Byzantine architectures
- **Allows read-only queries** to be performed locally within a site, the system continues serving read-only requests even in sites that are partitioned

Steward Protocol



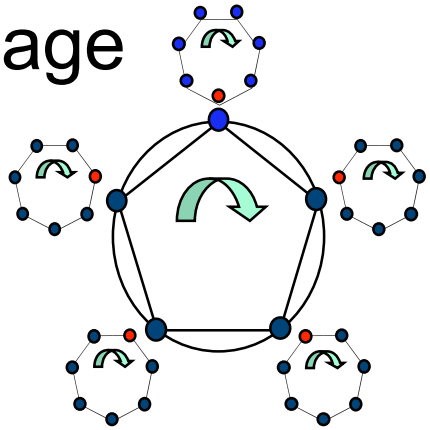
Messages are generated by sites

Each site has a representative

Global protocol has a leading-site

Global Protocol Details

- Uses a Paxos-like [Lam98,Lam01] protocol to mask site faults
  - Representative of the leading-site proposes order
  - Requires a majority of sites to have progress
  - If the leading site crashed, a new one is elected
- **Site fault**: considered crashed if a site is not able to generate a correct message (not enough majority), or gets disconnected



Local (Intra-site) Protocol Details

- Use BFT-like [CL99] protocols to mask replica faults:
  - Local representative coordinates the site protocol and forwards packets in and out of the site
  - Requires a proof that 2f+1 servers agreed on the order to ensure safety and local progress
  - Uses threshold digital signatures to ensure that no coalition of less than 2f+1 local Byzantine replicas can misrepresent the site on the wide area network
  - If the local representative fails, a new one is elected
- **Replica fault**: Byzantine fault

Ordering Updates

- Client sends update to local site
- Local site forwards update to leading-site
- Leading-side
  - Assigns local order (proposal for global ordering)
  - Propagates the proposal starting the acknowledgment phase
- Each site
  - Generates the acknowledgement using intra-site protocols
  - Orders when it saw a majority of acknowledgments from other sites
- Local site responds to client

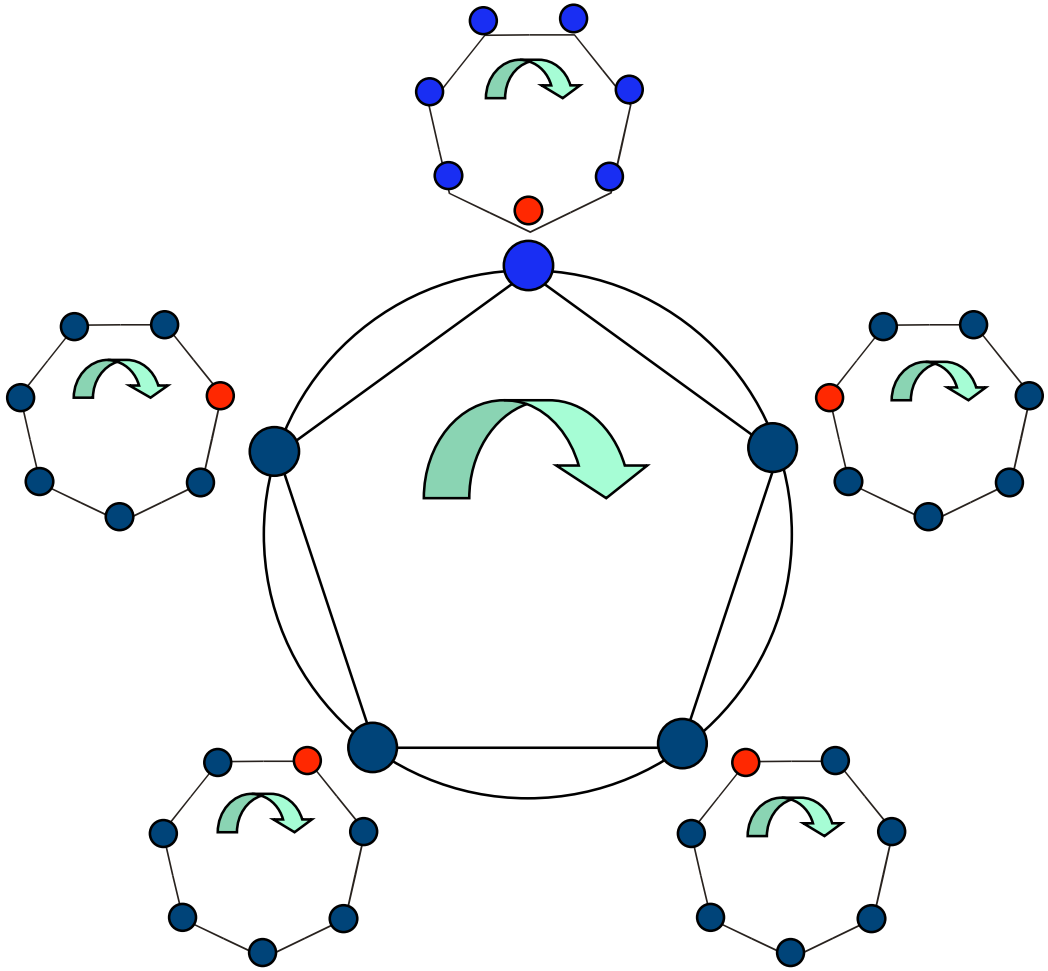
All messages are signed by the originators, messages that leave a site carry a threshold digital signature

The Devil is in the Details

- **Leader site and representative may fail**
- Select and change the representatives (change local view) and the leading-site (change global view), in agreement
- Transition safely between different leading-sites or representatives: reconciliation process
- Set timeouts to allow correct sites to have time to communicate

Coordination between Global and Local Representative Elections

- Local representative changed based on local timeout
- Leading-site representative changed based on a larger timeout allowing for communication with at least one **correct** representative at other sites
- Leading-site is changed after changing f+1 leading-site representatives



Reconciliation after a Local View Change

Goal: all correct local servers exchange information to make sure that they have enough information about pending Proposals to correctly enforce previous decisions

- New representative sends a sequence SEQ
- Every server sends a higher sequence SEQ_i representing updates he has ordered or acknowledged
- Representative collects **2f+1** responses, eliminates duplicates, selects update with highest view and broadcasts it to everybody, computes also the list of missed messages

Reconciliation after a Global View Change

Goal: all correct sites exchange information to make sure that they have enough information about pending Proposals to correctly enforce previous decisions

- New representative at leader site sends a sequence SEQ
- Every site sends a higher SEQ_i representing updates it has ordered or acknowledged
- Representative collects **f+1** responses, eliminates duplicates, selects update with highest global view and broadcasts it to everybody computes also the list of missed messages

Eliminate Malicious Nodes

- Sometimes we know which nodes are malicious
- Verifiable secret sharing allows us to detect the incorrect shares and the incorrect servers
- However, verification of the share is a relatively expensive operation
- Contradictory information sent to different participants

Optimistic Approach

We do not verify every partial signatures before combining

- Threshold digital signature verifies
  - The combiner can check that the signature is correct by using the public key. Proof for correctness and share verification are not needed in such a case
- Threshold digital signature does not verify
  - Detect which share(s) are incorrect: The combiner verifies the partial signatures
  - Malicious nodes partial signature eliminated
  - Potentially create a correct threshold signature by using other shares than the ones that were incorrect

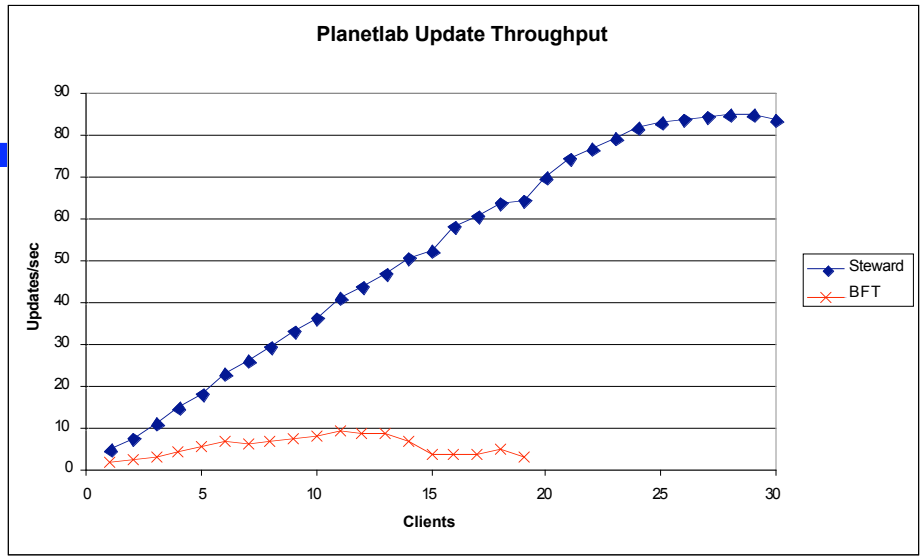
Putting it All Together

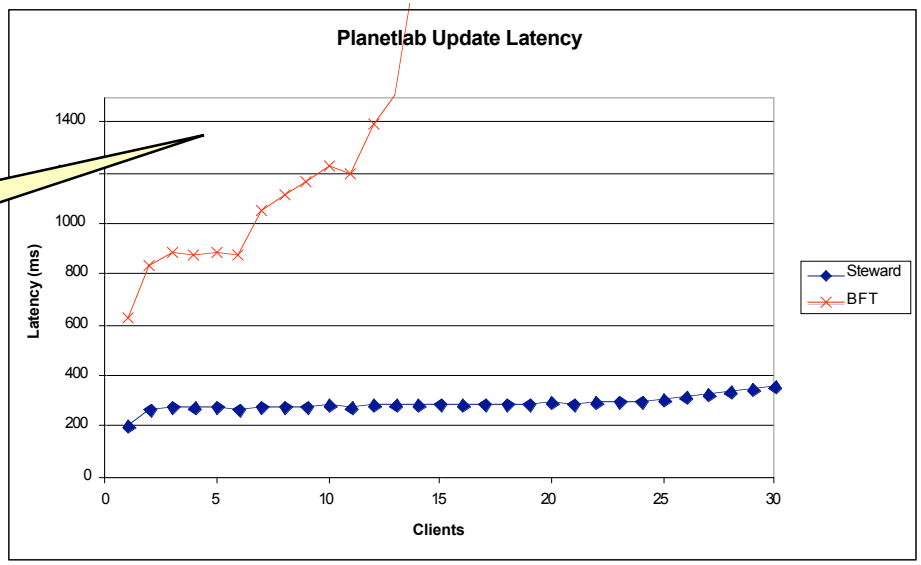
- Several protocols run in parallel
  - Order the updates
  - Intra-site representative election (or local view change)
  - Leading site election (or global view change)
- Reconciliation performed to transfer safely between views (either local or global)
- Can detect malicious nodes that contributed ‘wrong shares’ or has two-faced behavior

PlanetLab

- Selected 5 Planetlab sites, in 5 different continents: US, Brazil, Sweden, Korea and Australia.
- Measured bandwidth and latency between every pair of sites.
- Emulated the network on our cluster (20 machines, openssl), both for Steward and BFT.

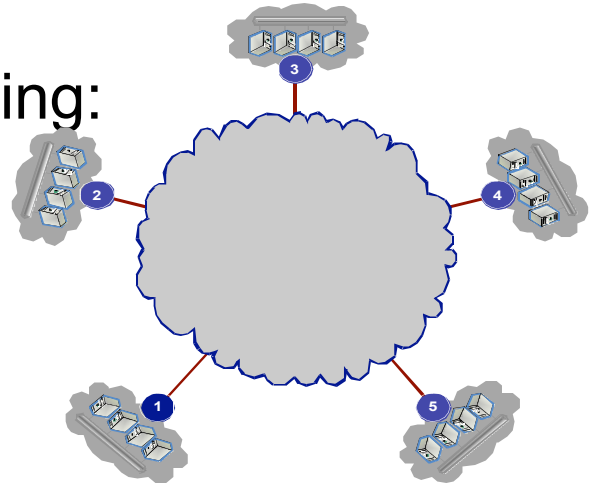
3-fold latency improvement.





Red Team: Attack Scenario

- Five sites, 4 replicas each.
- Red team had full control (sudo) over five replicas, one in each site.
- Compromised replicas were injecting:
  - Loss (up to 20% each)
  - Delay (up to 200ms)
  - Packet reordering
  - Fragmentation (up to 100 bytes)
  - Replay attacks



- Compromised replicas were running modified servers that contained malicious code.

Results

STEWARD WAS NOT COMPROMISED

- Safety and liveness guarantees were preserved.
- The system continued to run **correctly** under all attacks.

- Most of the attacks did not affect the performance.
- The system was slowed down when the representative of the leading site was attacked.
  - Speed of update ordering was slowed down to a factor of 1/5.
  - Speed was not low enough to trigger defense mechanisms.
  - Crashing the corrupt representative caused the system to do a view change and re-gain performance.

Lessons Learned (1)

- Using an hierarchical architecture improves scalability and availability
- Using an optimistic approach decreases the cost in no attacker case
- Eliminating malicious adversaries when possible improves convergence

Lessons Learned (2)

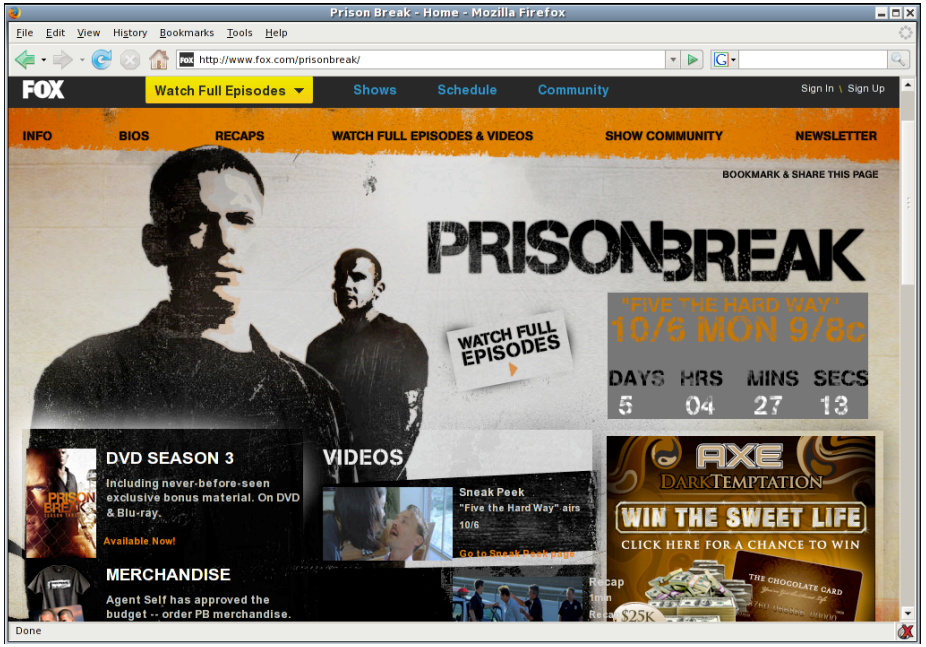
- Timeouts, timeouts, timeouts:
  - their setting is critical for protocol liveness (even more for a complex protocol)
  - they are also the vulnerable part that can be exploited by an attacker
- Complex protocols are more difficult to prove
- Models not sufficient to capture performance requirements (*how slow is slow progress?!*)

Unstructured Peer-to-peer Multicast Overlays with Byzantine Robustness

David Zage, Aaron Walters,
Cristina Nita-Rotaru

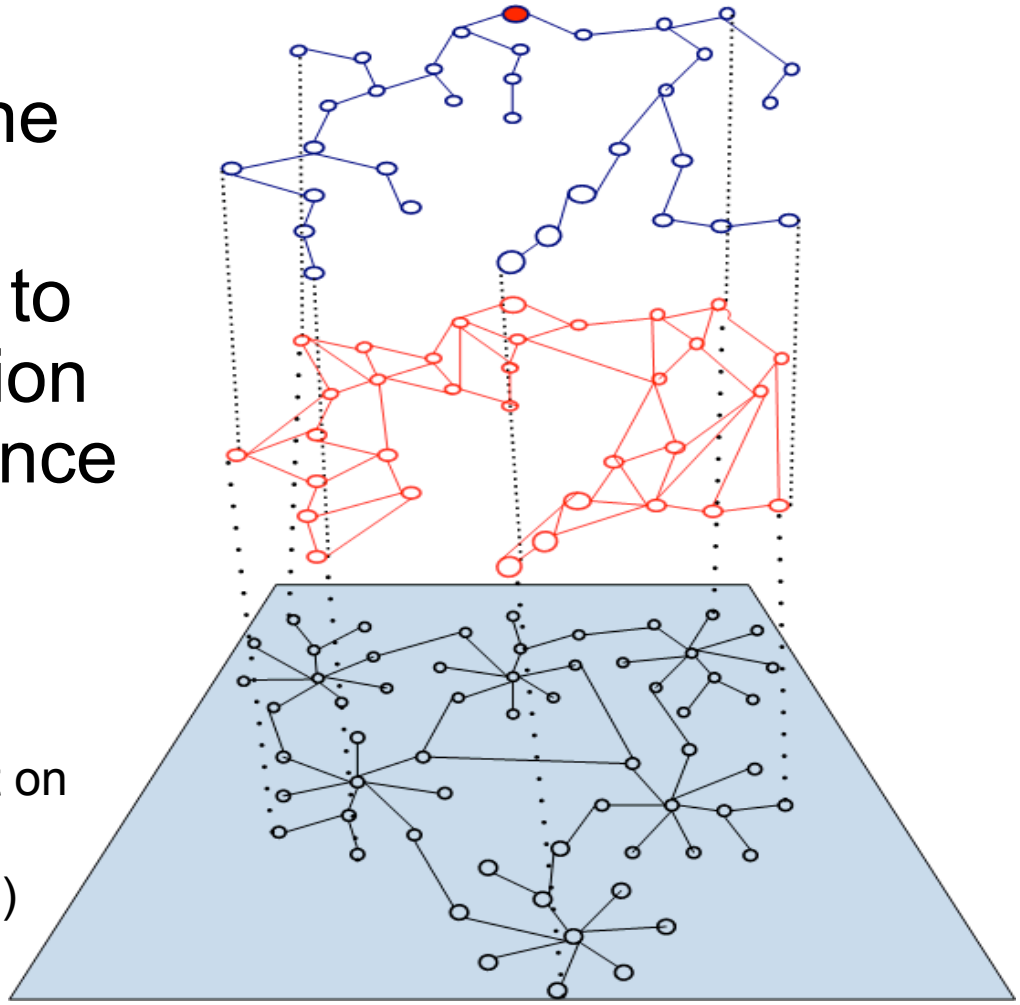
Streaming is a Growing Application

- TV companies are putting content online
- Enterprise networks are being used for video
  - Video streaming accounts for 7% of bandwidth
  - P2P video applications are found on 43% of networks



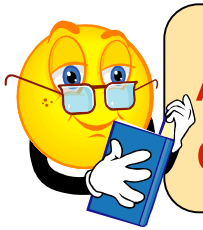
Unstructured Multicast Overlay

- Mesh control plane
- Tree-based multicast: adapts to maintain application specific performance
- Each node maintains:
  - Parent
  - Peer set: no constraint on neighbor selection
  - Routing table (children)



Adaptation

- Nodes collect metrics and compute a utility function:
  - Passive observation: their own performance
  - Periodic probing: performance of peer nodes
- A node change its parent in the tree based on the utility function
- ESM metrics: **available bandwidth, latency, RTT and saturation degree**



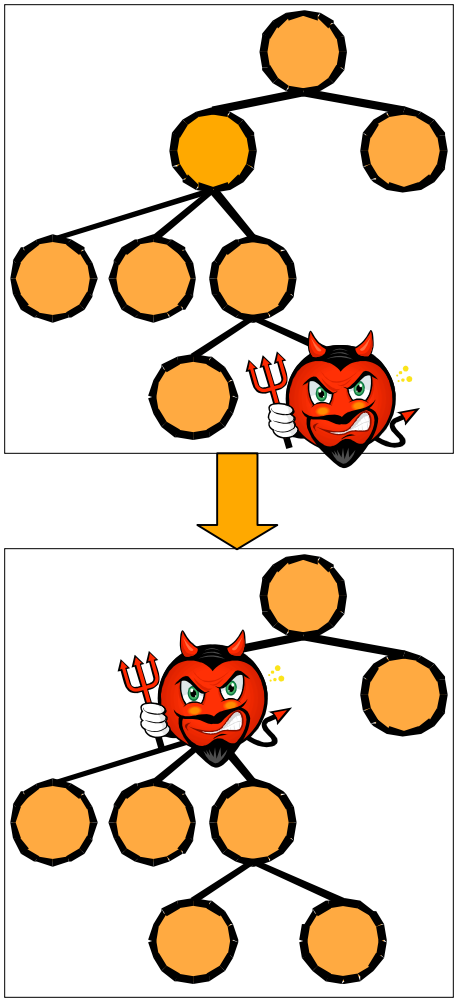
Accurate interpretation of performance observations and the correctness of the responses from probed nodes are critical!

Attacks Exploiting Adaptation

- Compromised nodes: **lie about the observation space (bandwidth, latency, degree)**
- Classification of attacks based on their effect on the control of path:
  - Attraction attacks
  - Repulsion attacks
  - Disruption attacks
- Used to facilitate further attacks:
  - Selective data forwarding
  - Traffic analysis
  - Overlay partitioning
  - and more ….

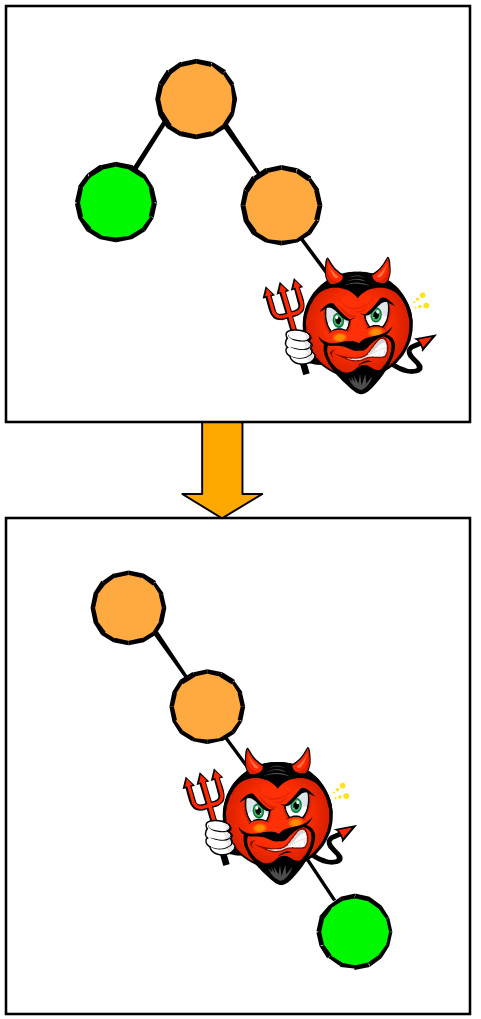
Attraction Attacks

- *The more children a node has or higher in the tree is, the higher the control of data traffic*
- **Attacker goal**: attract more nodes as children in the overlay structure
- **How does it work:** a node **makes things look better** by lying about its reported metrics
- **Result:** controlling significant traffic, further conduct traffic analysis or selective data forwarding

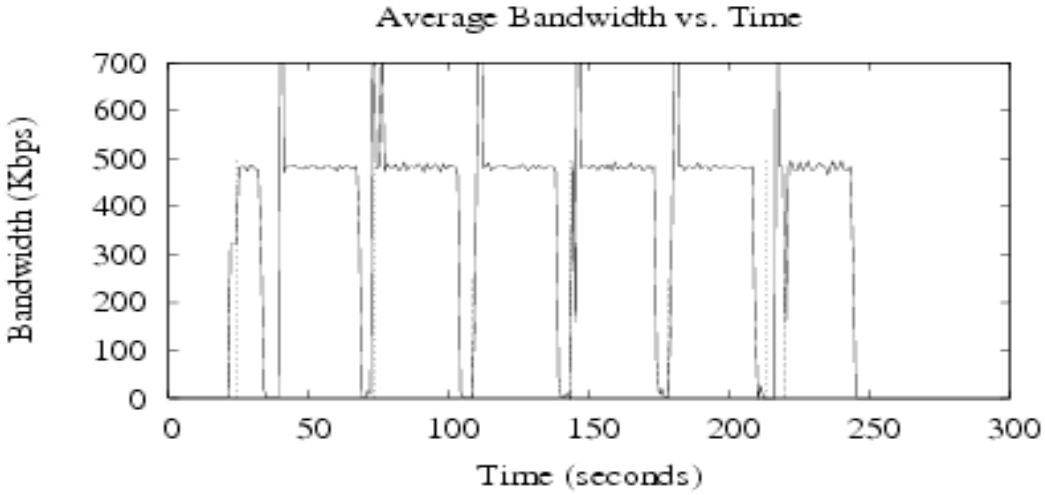


Repulsion Attacks

- *A node in the overlay may affect the perception of the performance from the source*
- **Attacker goal**: reduce the appealing of other nodes or its own
- **How does it work**:
  - a node lies in responses to probes
  - a node manipulates the physical or logical infrastructure to create the perception of lower utility of other nodes
- **Result**: freeloading, traffic pattern manipulation, augmenting attraction attacks, instability



Disruption Attacks



- *Frequent adaptations can create instability*
- **Attacker goal**: exploit the adaptation to turn the system against itself
- **How does it work:** attacker injects data to influence the observation space metric data to generate a series of unnecessary adaptations, similar with TCP attack
- **Result**: jitter, flapping, or partitioning the overlay

Reduce the Poor Adaptations

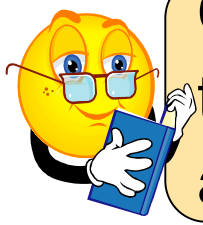
- Reduce the likelihood of making poor adaptations, *before* they take place
- Use spatial and temporal correlations based on statistical outlier detection to filter out outliers
- **Challenges**: Analyzing the effect on overall performance, method will not completely eliminate bad adaptations

To Avoid Detection A Node Must Lie:

- C1: consistently with what the other peers are reporting during a probe cycle about current conditions
- C2: consistently with the bandwidth, latency, and influence yielded towards the RTT
- C3: consistently with what it said in the past.

Local Outlier Detection

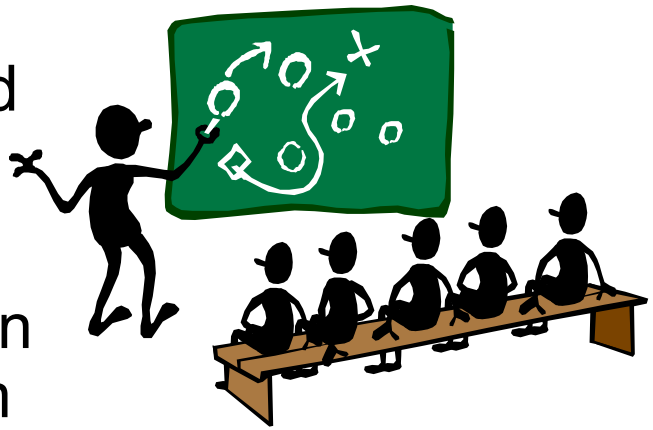
- **Spatial outlier detection** compares the reported metrics received from each node in the set of probed nodes (C1 and C2).
- **Temporal outlier detection** examines the consistency in the metrics received from an individual probed node over time (C2 and C3).



Outlier: data point that is significantly different from the rest of the data in the observation space based on a measure of **distance**

Spatial Outlier Detection

- Features: bandwidth, latency, and RTT
- Done during each probing period
- Observation tuples used to compute the centroid
- Compare how far the observation tuple for each node is away from the centroid.



Spatial outlier detection compares the reported metrics received from each node in the set of probed nodes

Temporal Outlier Detection

- Temporal centroid: mean, standard deviation, and sample count associated with the observation tuple for each of the peers
- Nodes do not need to maintain all history, centroid is incrementally updated with observations received during each probe cycle

Temporal outlier detection compares the metrics received from an individual probed node over time

Experimental Setup

- Using ESM
- Planetlab and DETER
- Deployments of 100 nodes
- Experiment durations of 30 and 90 minutes.
- Saturation degree of 4-6 nodes
- Constant bit rate of 480 Kbps

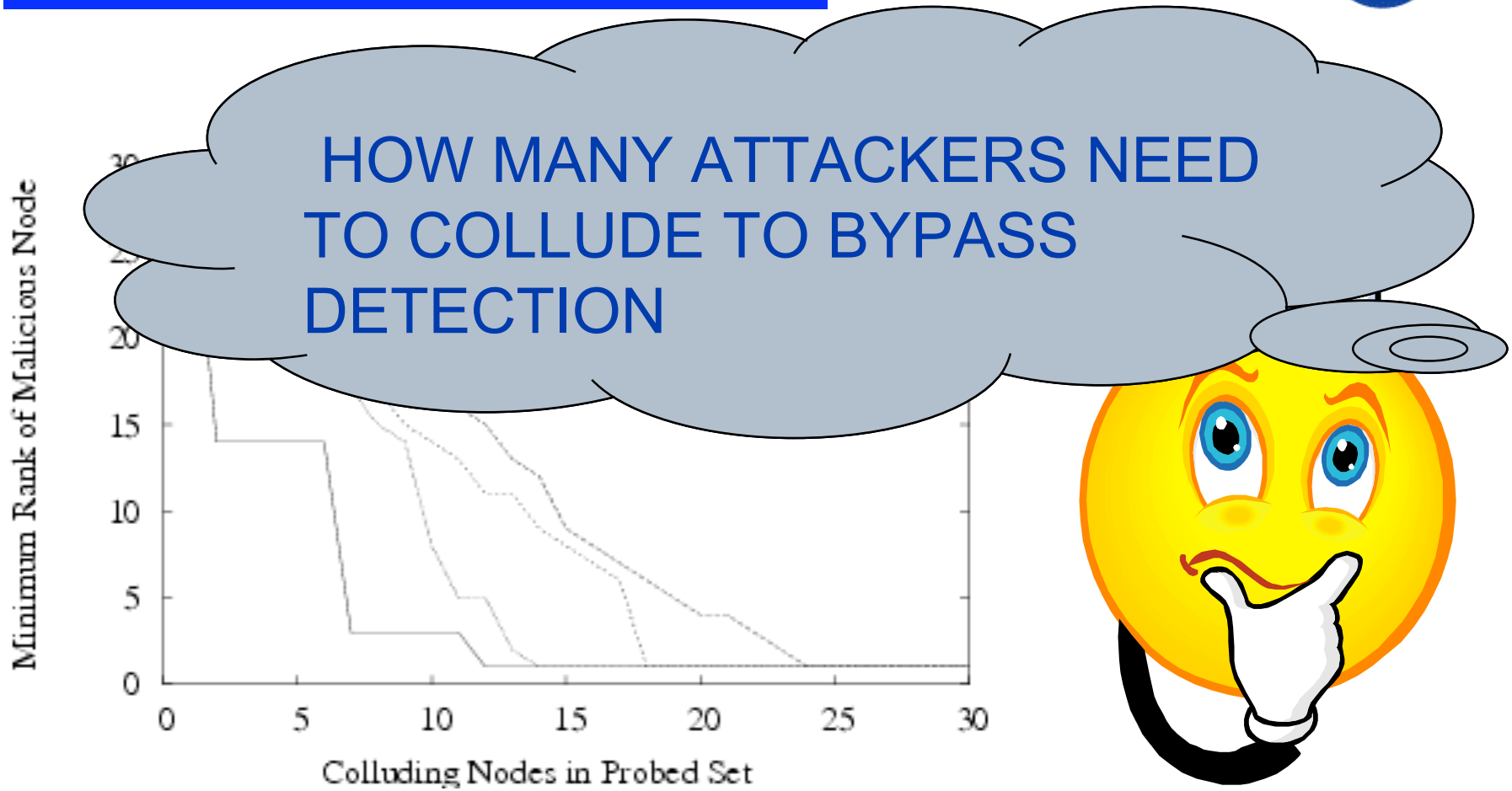
Effectiveness of Outlier Detection

- 100 nodes, over 60 minutes, 30% malicious nodes

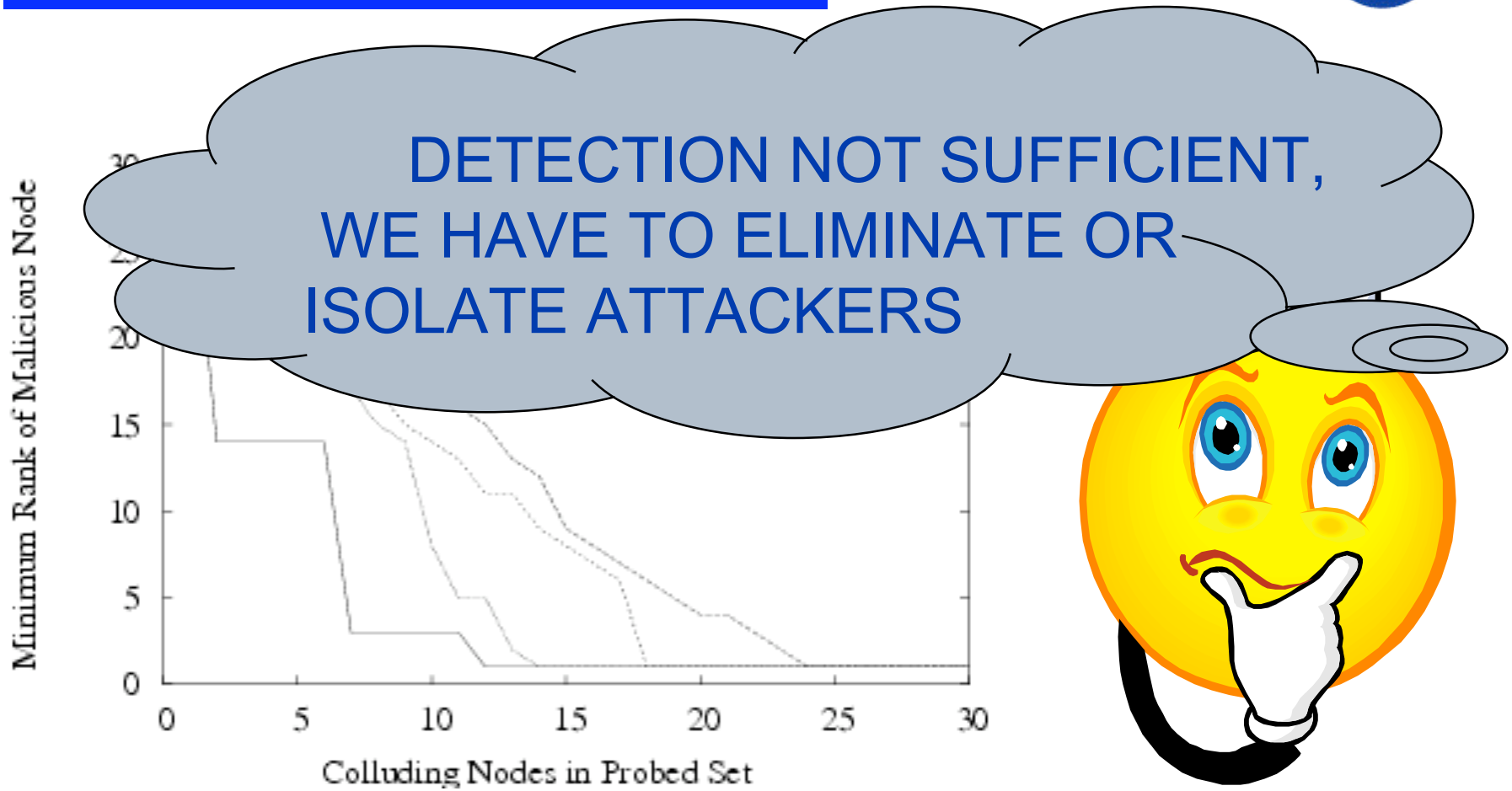
Experiment	Changes to Malicious Parent	Total Parent Changes
No Lying	5	833
Lying	172	1032
Spatial	70	800
Spatial/Temporal	35	604

Improves stability and reduces the number of malicious changes (bandwidth did not change, with less changes)

Resilience to Coalition of Attackers



Resilience to Coalition of Attackers

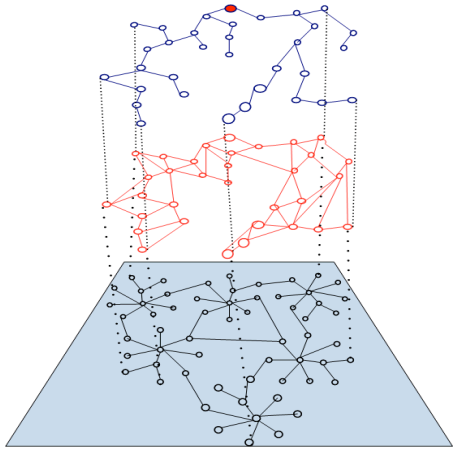


Isolating Malicious Nodes

- Neutralize malicious nodes once detected
  - Improves performance
  - Outlier detection does not “learn” malicious behavior
- Two-pronged approach
  - Local suspects list for quick response
  - Global black list created from shared information

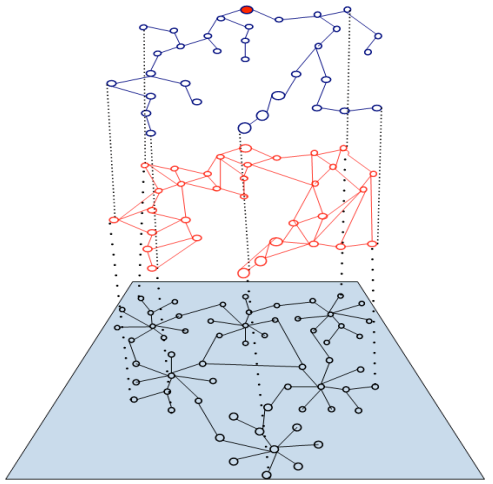
Local Response

- Every node suspects a neighbor based on how far it was from the spatial and temporal centroids
- Suspect list is gossiped to local neighbors
- Good behavior rewarded (compensates also for transient network conditions)
- If a ‘suspect’ reaches a threshold suspicion value, it is reported to the source



Global Response

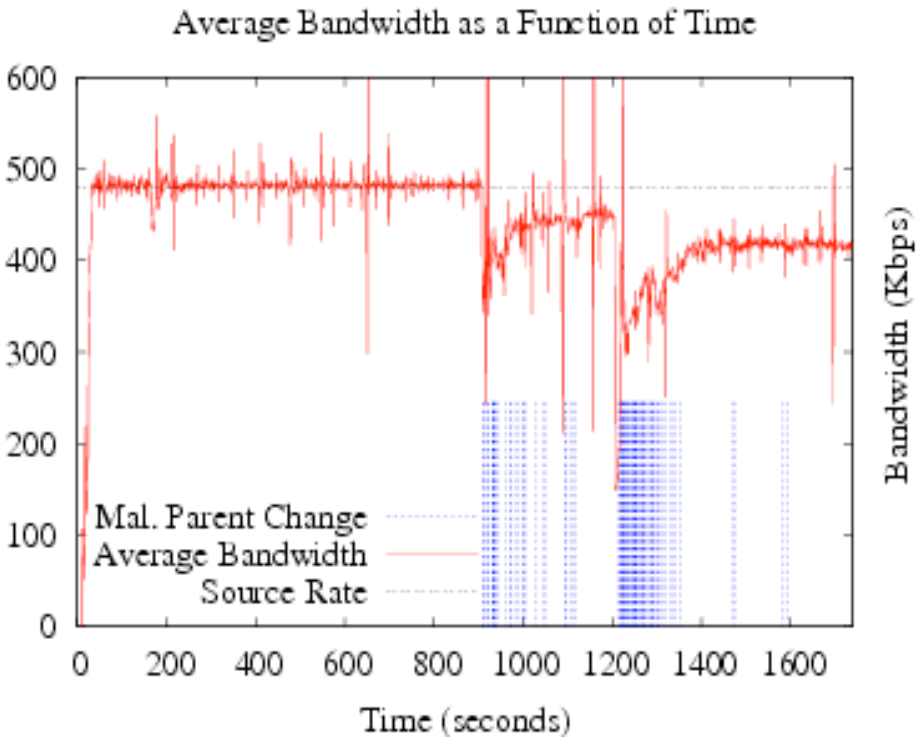
- Source aggregates local suspect list into global view of trust
- Adaptation of the EigenTrust [KSG03] reputation system
- Nodes falling below a threshold are placed on a global black list which is periodically disseminated to all nodes



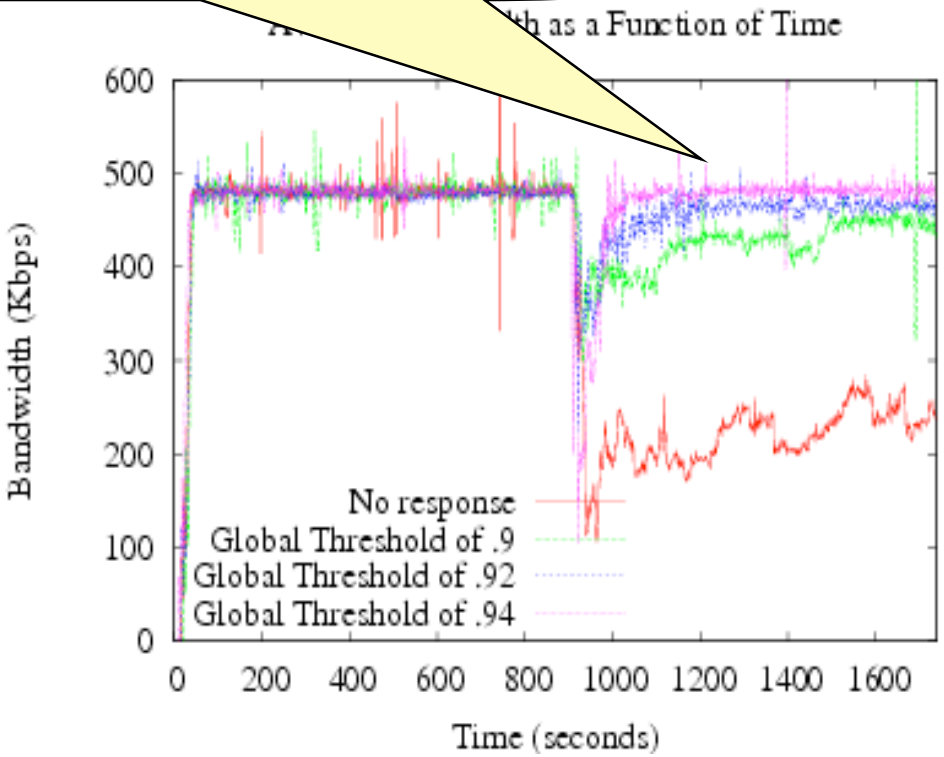
Effectiveness of Response Mechanism

- 100 nodes over 60 minutes, 30% malicious nodes

Bandwidth returns to value before attack

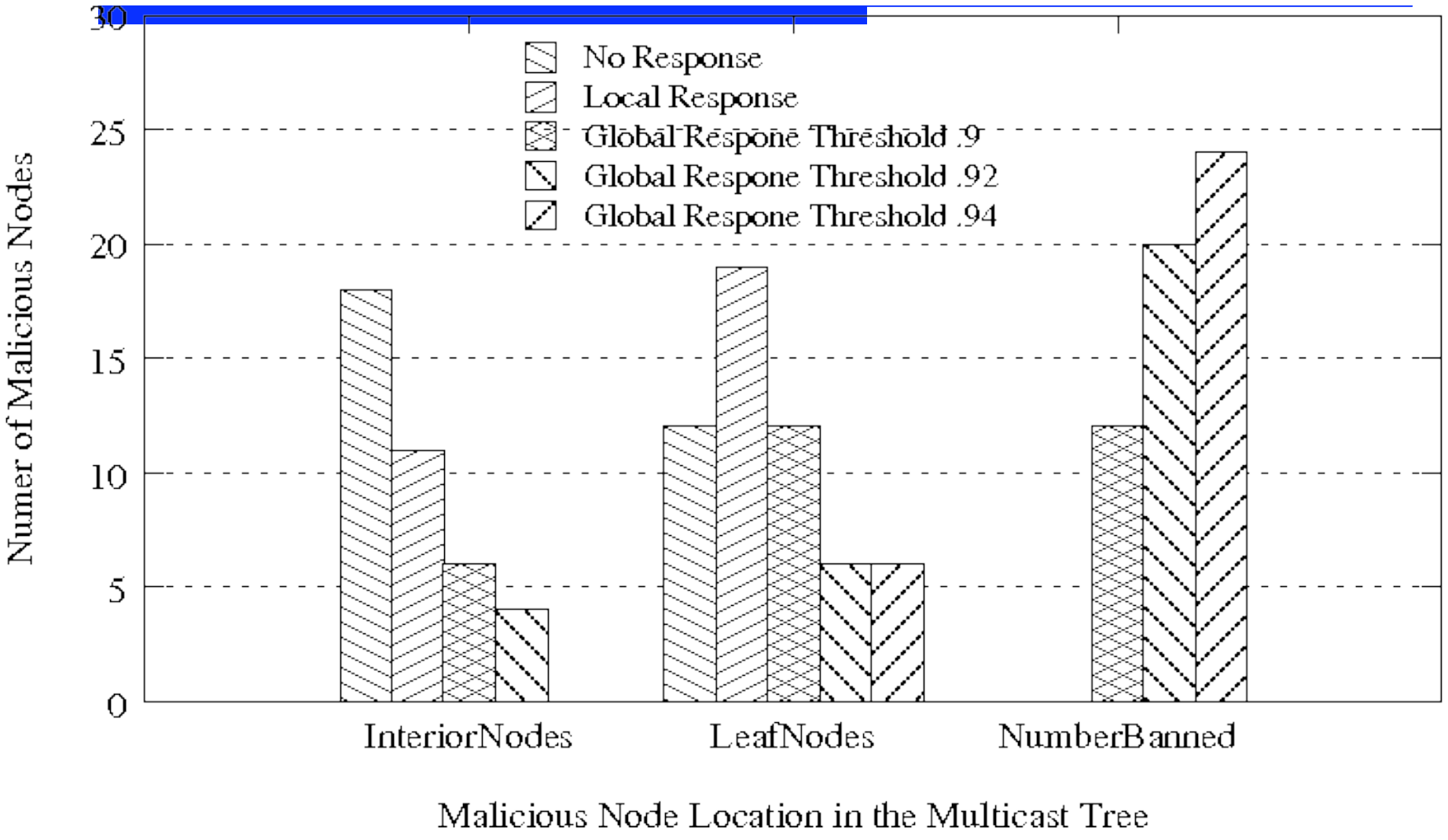


Local response only



Local and Global response

Malicious Nodes Pushed as Leaves or Banned



Lessons Learned

- Detection not sufficient, attackers must be isolated
- Isolation should be done carefully to avoid disconnecting the network
- Control-plane defenses should be combine with data-plane feedback
- It comes down to reputation

Thank You!

crisn@cs.purdue.edu

http://www.cerias.purdue.edu/homes/crisn