Scaling Byzantine Fault-tolerant Replication to Wide Area Networks

Cristina Nita-Rotaru
Dependable and Secure Distributed Systems Lab
Department of Computer Science and CERIAS
Purdue University

OUR TEAM

- Johns Hopkins University: *Yair Amir, Claudiu Danilov, Jonathan Kirsch, John Lane, Jonathan Shapiro*
- The Hebrew University of Jerusalem: Danny Dolev
- Purdue University: *Cristina Nita-Rotaru, Josh Olsen, David Zage*
- Funded by DARPA Self-Regenerative Systems DARPA Program and NSF CyberTrust Program

http://www.cerias.purdue.edu/homes/crisn/lab/scalable.html

Applications of Distributed Services

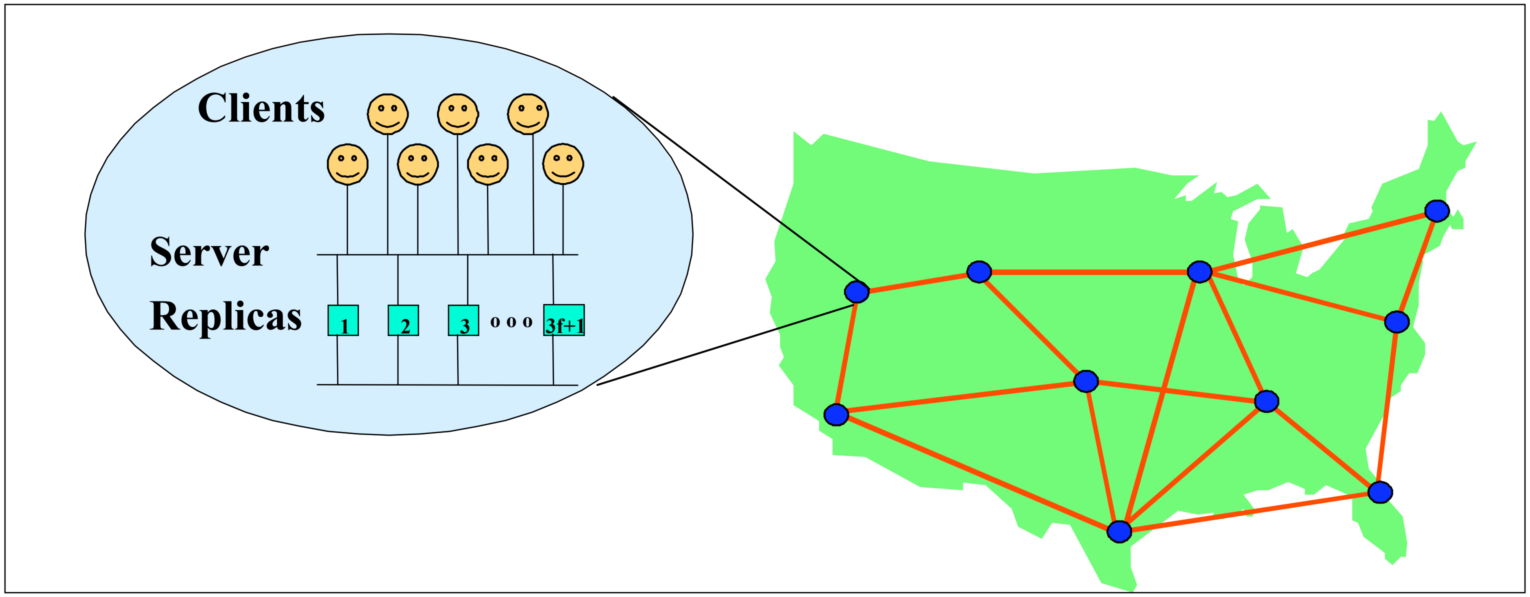
Distributed systems provide support for data availability, fast access to information, and efficient communication between multiple parties.

- Software solutions for high-availability clusters
- Distributed monitoring
- Collaborative applications
- Databases for national emergency systems
- Network centric warfare applications

Network Centric Warfare Applications

- Operate on WANs settings: unreliable channels and intermittent network connectivity
- Require timely decisions based on available information, although information may not be 'the most recent and consistent'
- Critical information is often not large.
- Every piece of information is usually generated by a unique source.

Deconstructing Distributed Services



- A service is **replicated** by several servers that **coordinate** to serve clients
- Clients issue requests to servers, then wait for answers
- Communication achieved via message passing
- Goals: respond fast, avoid inconsistencies

What Can Go Wrong?

- Benign faults :
  - Clients and servers can crash
  - Network can physically partition or experience high delays
- Outside attackers (not part of the system):
  - Eavesdrop communication
  - Impersonate participants, inject/modify/replay messages
- Inside attackers (compromised servers or clients):
  - Stop behaving correctly, for example: clients can inject malicious data, servers do not process requests, do not forward data correctly

How to Defend?

- Detection:
  - intrusion detection systems
- Prevention:
  - access control or firewalls
  - proactive security
- Mitigation (Byzantine fault-tolerance):
  - provide service to correct participants even if several participants are corrupted
- The above methods do not exclude each other and can be used to cover each other's limitations.

This Talk

Focuses on mitigating insider attacks, by presenting the first Byzantine fault-tolerant replication system that makes Byzantine replication practical in wide area networks.

Outline

- Introduction
- **Overview of state-of-the-art solutions**
- Our approach: Steward
- Experimental results
- Red-team experiment
- Summary

Revisiting Consensus…

- As any server can answer requests, for consistency, all servers "must agree"
- Synchronous communication does not model real networks
- Asynchronous communication, agreement (consensus) can not be reached even if there is just one benign fault [FLP83].

Circumventing [FLP83]

Avoid agreement between all parties, use quorum systems

Make the guarantees probabilistic with the use of randomization

Maintain safety, sacrifice liveness

State Machine Replication [Sch90]

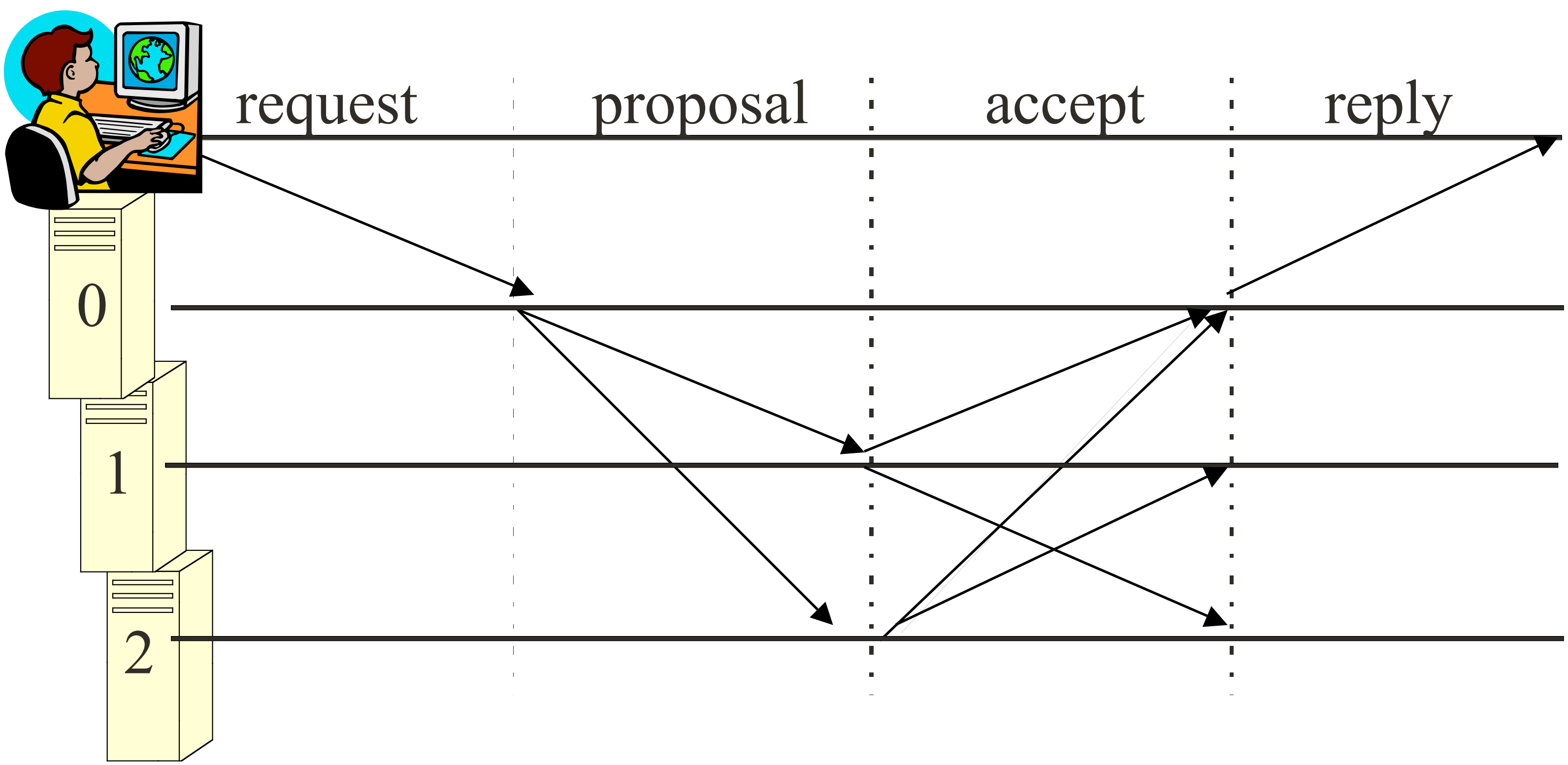
- Relies on one server (leader) to coordinate the agreement on order in which requests are processed
- If the leader dies, a new leader must be selected to ensure progress!

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 for a client

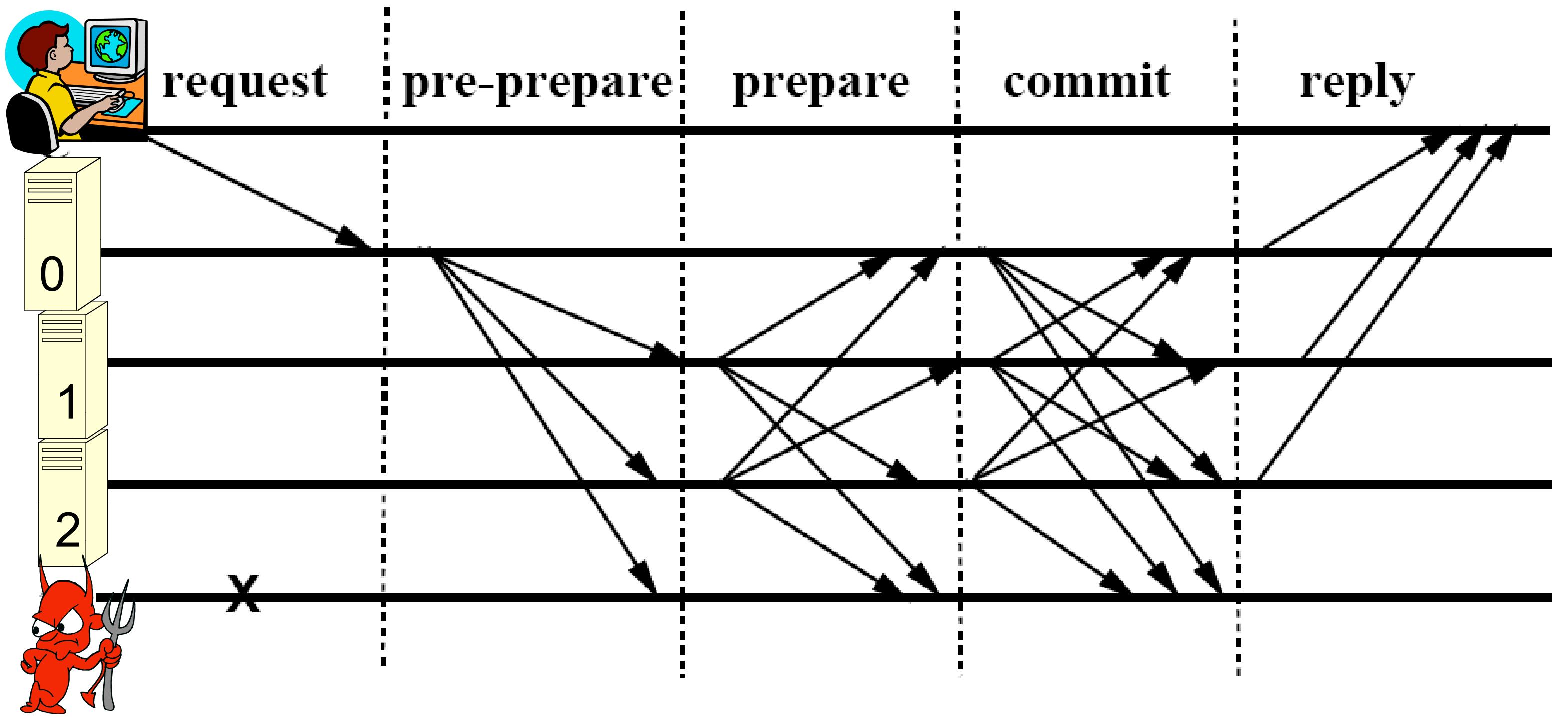
Byzantine faults: BFT [CL99]: must contact 2f+1 out of 3f+1 servers and uses 3 rounds to allow consistent progress,
f+1 identical answers needed by a client

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



f servers can crash, f=1 in this example

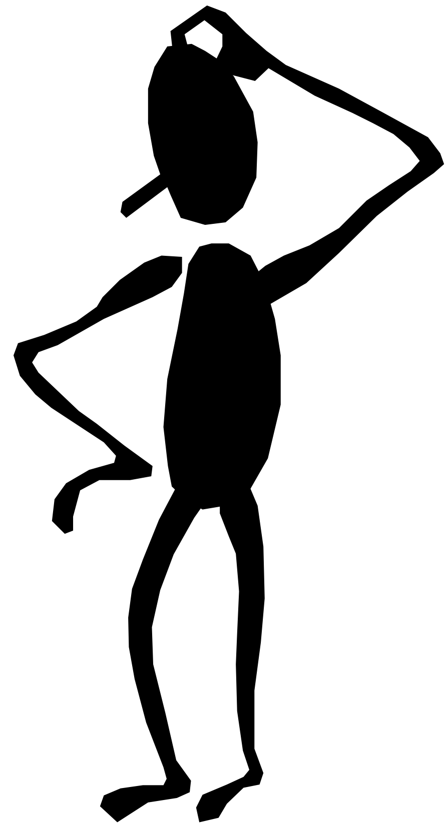
State-of-the-Art: BFT [CL99]



f servers are malicious, f=1 in this example

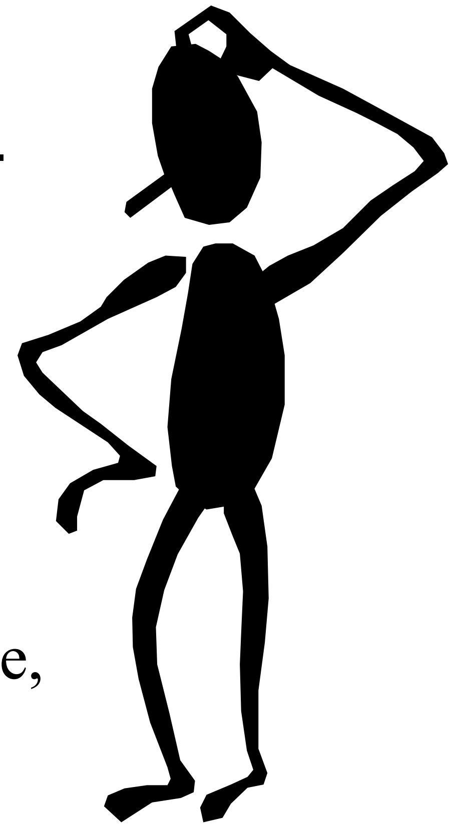
The Problem with Byzantine Servers …

- We can not trust the information reported by any server
- We can not delimitate correct behavior from an incorrect one in all cases
- We can not have a solution when more than a tolerate threshold of nodes collude



Limitations of Current Solutions

- Limited scalability due to **3 round all-peer** exchange
- Strong connectivity requirements
  - **2f+1 (out of 3f+1) to allow progress** and f+1 for the client to obtain a correct answer
  - On WAN: Partitions are a real issue, clients depend on remote information, long delays



Outline

- Introduction
- Overview of state-of-the-art solutions
- **Our approach: Steward**
- Experimental results
- Red-team experiment
- Summary

Our Solution: Steward

- **Our solution: use a hierarchical architecture**
  - Every site acts as one entity that can only crash if assumptions are met
  - Run fault-tolerant protocols between sites
  - **Result**: less messages and one communication round less in wide area networks
- Other approaches:
  - Minimize cost in fault-free [RAS04] (UIUC)
  - Probabilistic guarantees [SINTRA](IBM Zurich)

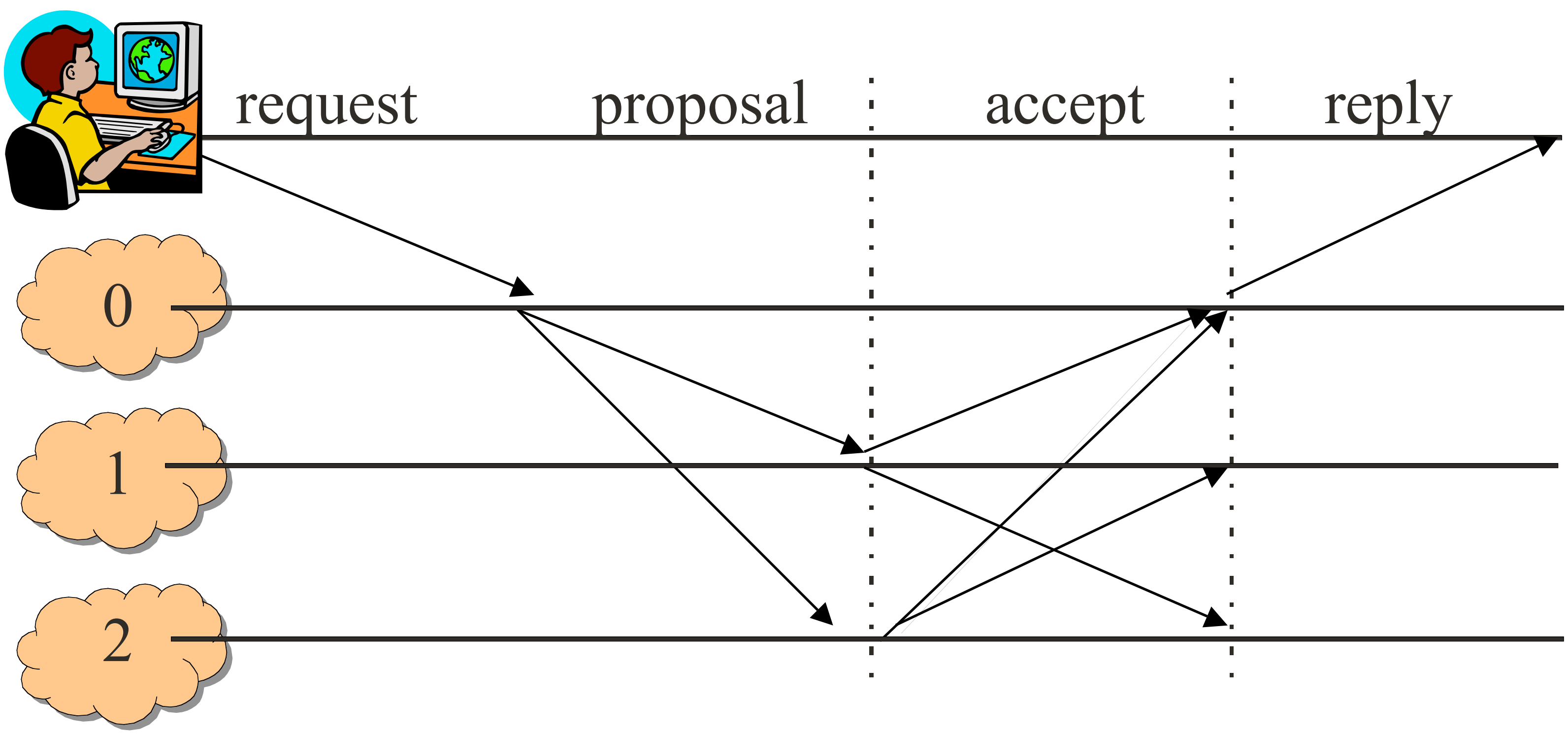
Advantages of Steward Architecture

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

IT REQUIRES MORE HARDWARE, each site has 3f+1 servers

As any other Byzantine protocols, assumes independent failures

Steward Wide Area Protocol



0, 1, 2 are SITES and not SERVERS

Wide-Area Protocol Details

- One of the sites act as the leader (associated with a global view) in the wide area protocol
- The representative of the leading site is the one assigning sequence numbers to updates
- Requires messages from a majority of sites to have progress
- If a site is not able to generate a correct message (not enough majority), or gets disconnected, the site is perceived as crashed

Intra-site Protocol

- Use BFT-like [CL99, YMVAD03(UTAustin)] protocols to mask local Byzantine replicas:
  - A representative (associated with a local view) coordinates the protocol and forwards packets in and out of the site
  - Requires at each step a proof that 2f+1 servers agreed on the order to ensure safety (ensures that any 2 sets of 2f+1 will intersect in a correct replica)
  - When the representative fails, a new one is elected
- Use threshold digital signatures to ensure that local Byzantine replicas cannot misrepresent the site on the wide area network.

Threshold Digital Signatures

Threshold digital signatures allow N entities authenticate a message by generating one signature s.t. any k entities can create a valid signature, but k-1 cannot.

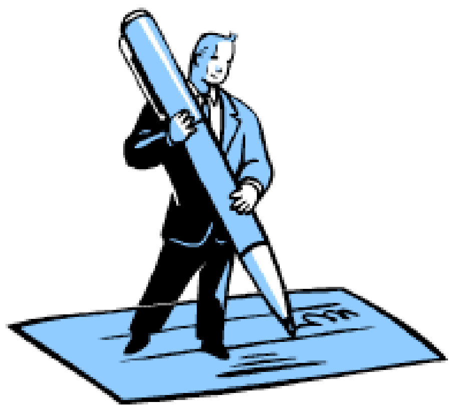
- Our choice is the RSA threshold signature proposed by Shoup in [Sch99]
- Generating a threshold signature requires a distributed protocols
- Verification is similar to RSA verifying
- Provides verifiable secret sharing

Threshold RSA Share Generation

- A trusted dealer generates the public (e,n) and private (d) RSA keys and then splits the private key d to **N shares**, s.t any **k out of N are enough to reconstruct the secret**.
- Select randomly a polynomial with a k-1 degree (as in Shamir's secret sharing).
- The dealer computes individual shares s_i.
- Dealer creates verification proof (involves modular exponentiation).
- More expensive than regular RSA, requires also safe primes.

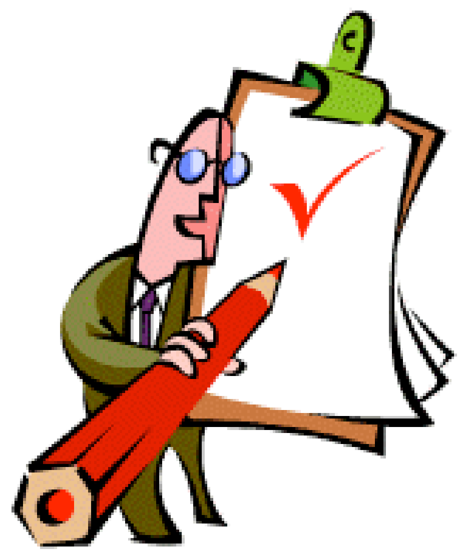
Generating a RSA Threshold Signature

- Each entity owns a share s_i:
  - Computes its individual signature and a proof of correctness (based on individual shares and verification proof).
  - Sends the individual signatures and the proof of correctness of the signature to the combiner.
- The combiner:
  - Collects all individual signatures.
  - Verifies that they were generated using the shares from the initial secret that was split (using the proof of correctness)
  - Generates the threshold signature.
- Much more expensive than one regular RSA signature.

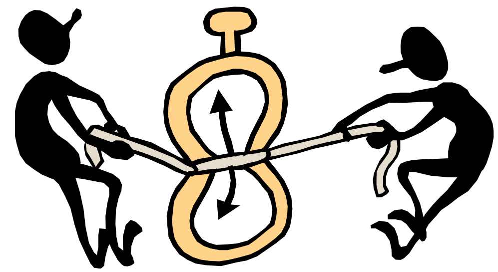


Verifying a RSA Threshold Signature

- Anybody can verify the signature based on the public key.
- Computation cost similar with to a single regular RSA digital signature verification.
- Consequence:
  - Remote sites only need one public key per site.



Ordering Updates



- Client sends update to local site
- Local site forwards update to leader site
- Leader side
  - assigns local order using BFT-like, then threshold signing - acts as proposed ordering for global protocol
  - propagates the proposal for global ordering 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

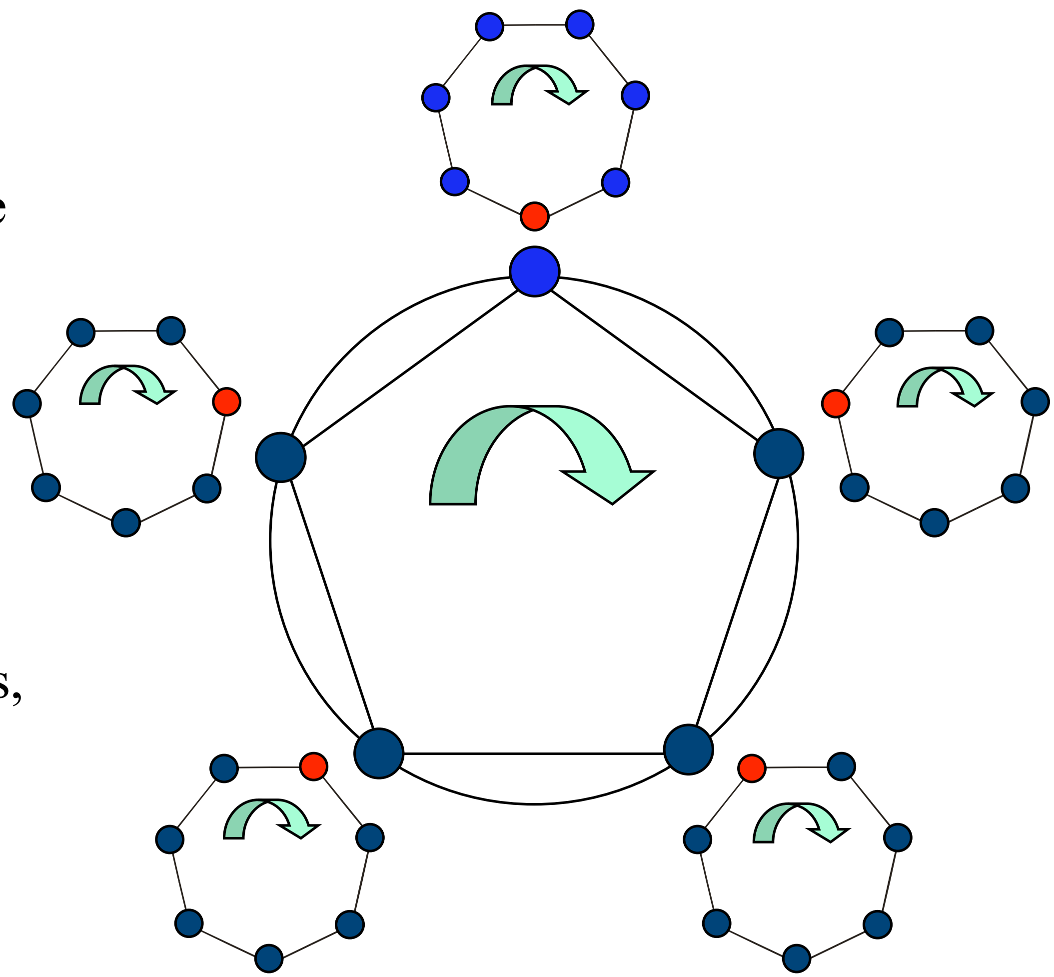
All messages are signed by the originators, messages that leave a site carry a threshold digital signature

The Devil is in the Details

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

Representative/Leader-Site Election

- Sites change their local representatives based on timeouts
- Leader-site representative has a larger timeout allowing for communication with at least one **correct** representative at other sites
- After changing f+1 leader-site representatives, servers at all sites stop participating in the protocol, and elect a different leader-site



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

- We can not always know which nodes are malicious.
- Use verifiable secret sharing:
  - If the threshold signature does not verify, then some partial signatures were not correct
  - Verifiable secret sharing allows us to detect the incorrect shares and the incorrect servers
- The drawback: verification of the share is a relatively expensive operation

Eliminate Malicious Nodes: Our 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 nodes that contributed 'wrong shares'

Outline

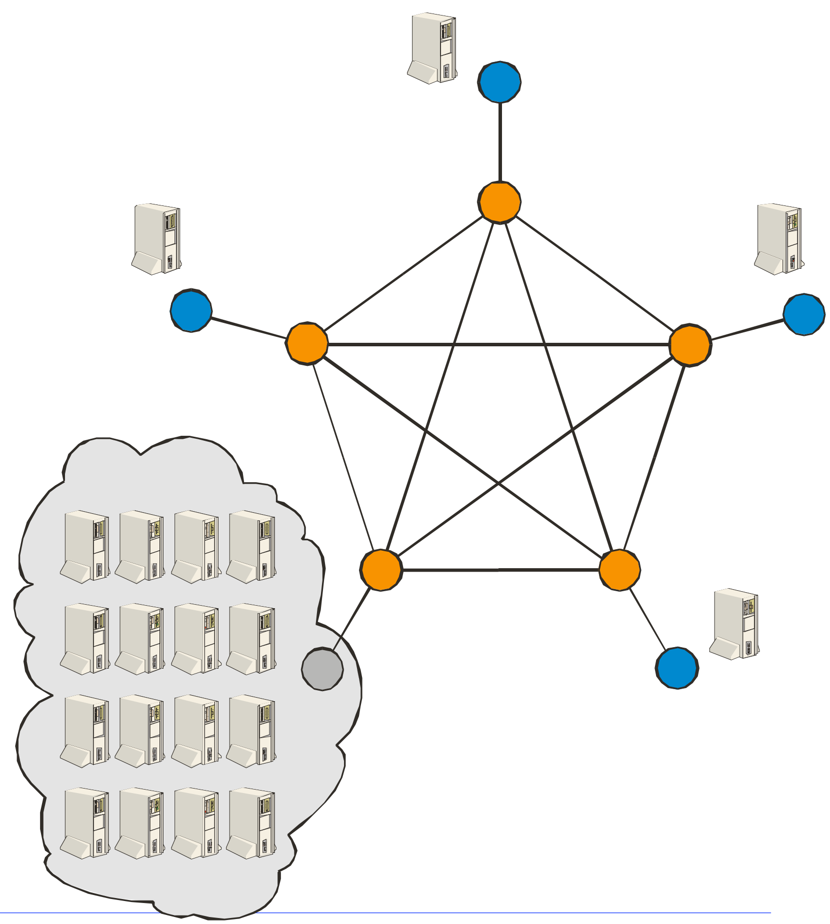
- Introduction
- Overview of state-of-the-art solutions
- Our approach: Steward
- **Experimental results**
- Red-team experiment
- Summary

Testing Environment

- Platform: Dual Intel Xeon CPU 3.2 GHz 64 bits, 1 GByte RAM, Linux Fedora.
- Cluster of 20 machines
- Our own threshold crypto library, uses Openssl
- Baseline operations:
  - RSA 1024-bits sign: **1.3 ms,** verify: **0.07 ms**.
  - Perform modular exponentiation 1024 bits, ~**1 ms**.
  - Generate a 1024 bits RSA key ~55ms.

Case 1: Symmetric Network

- Synthetic network used for analysis and understanding.
- 5 sites connected with equal bandwidth/latency links,
- 50 ms wide area links between sites.
- One fully deployed site of 16 replicas; the other sites are emulated by one computer each.
- Varied wide area bandwidth and the number of clients.



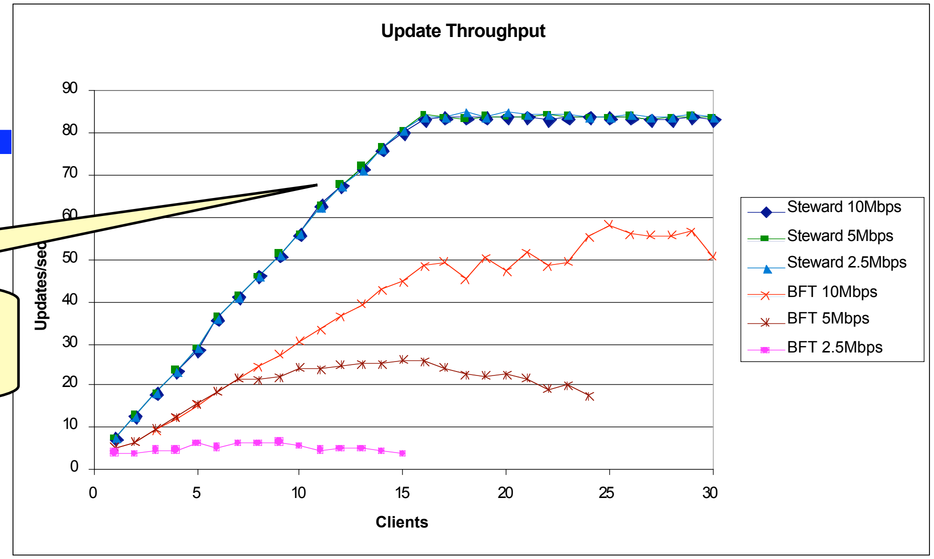
Write Update

Steward:

- 16 replicas per si...

sites are emulated).

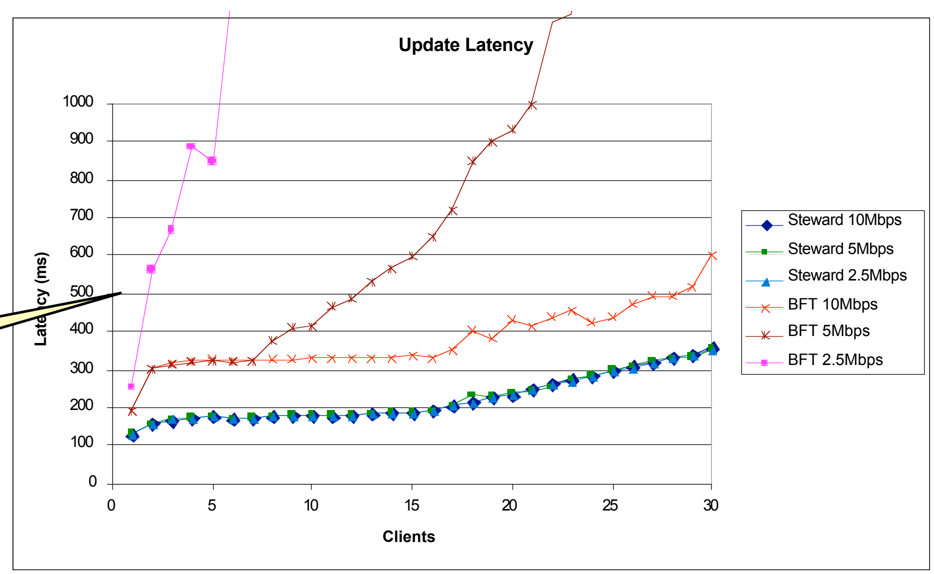
Steward achieves higher throughput



BFT:

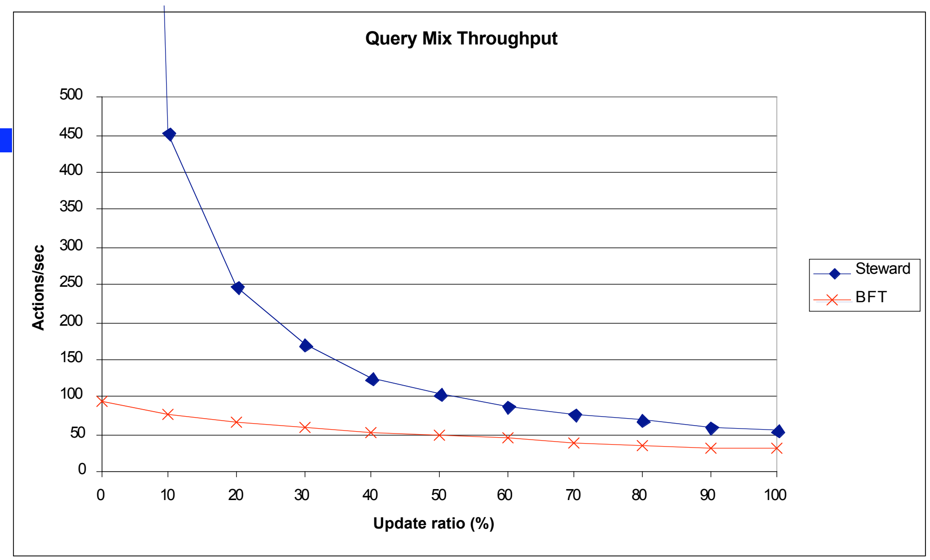
- 16 replicas total.
- 4 replicas in one site, 3 replicas in each other site.
- Update only performance (no disk writes).

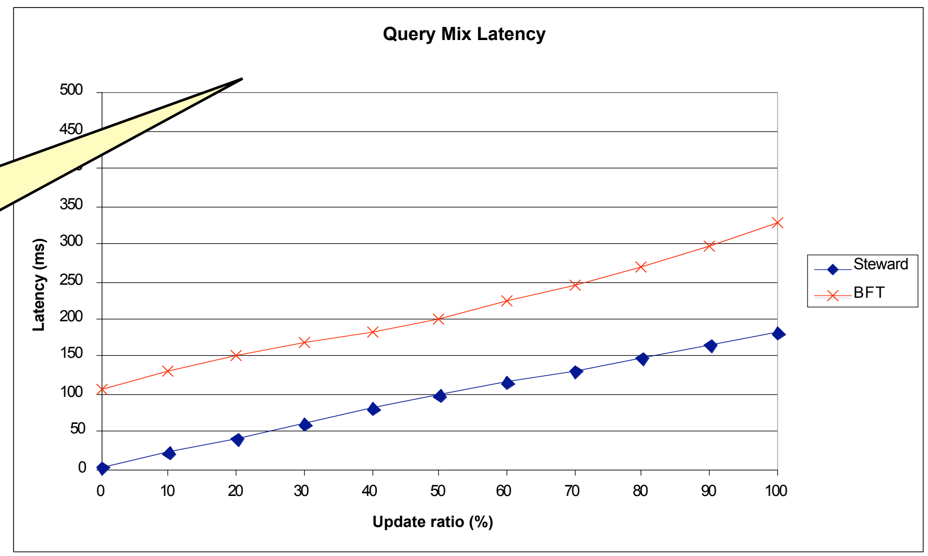
BFT latency goes to the roof



Read-only Query

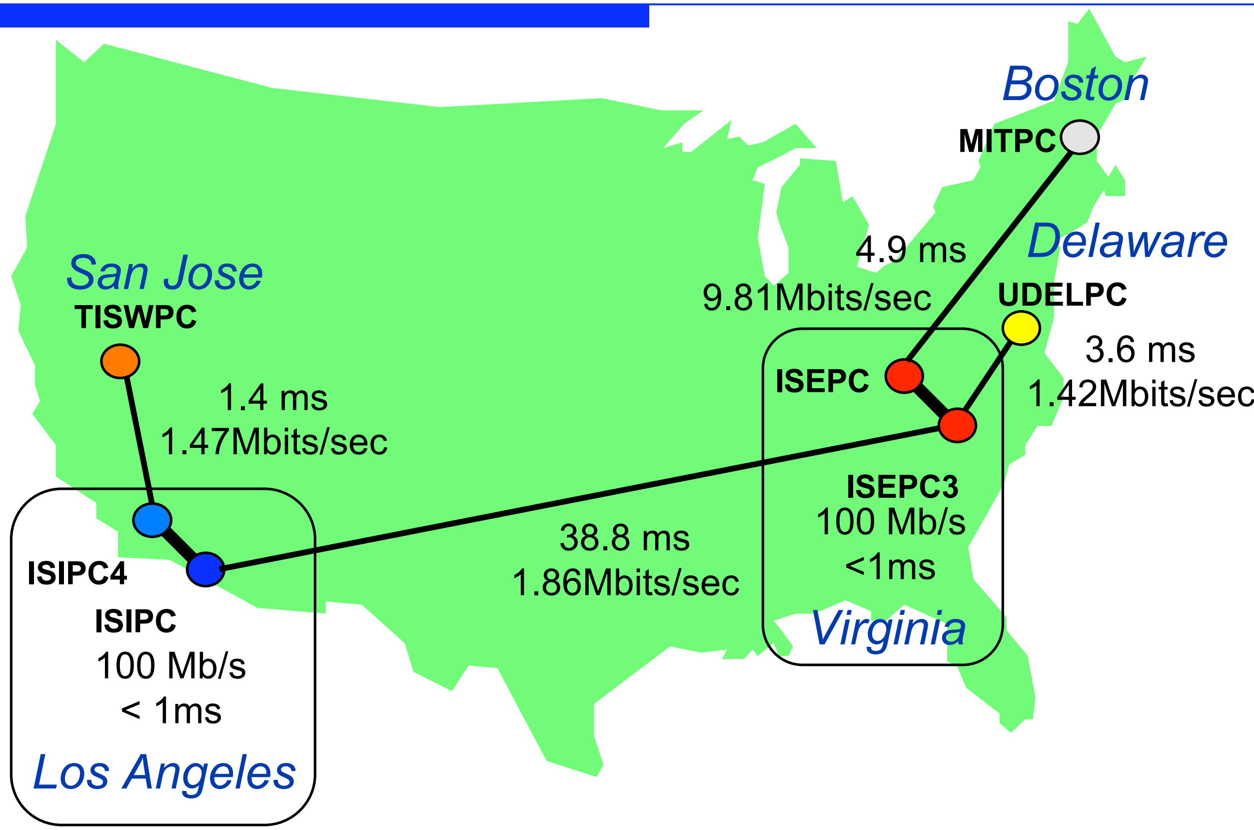
- 10 Mbps on wide area links.
- 10 clients inject mixes of read-only queries and write updates.
- None of the systems was limited by bandwidth





Performance improves between a factor of two and more than an order of magnitude.

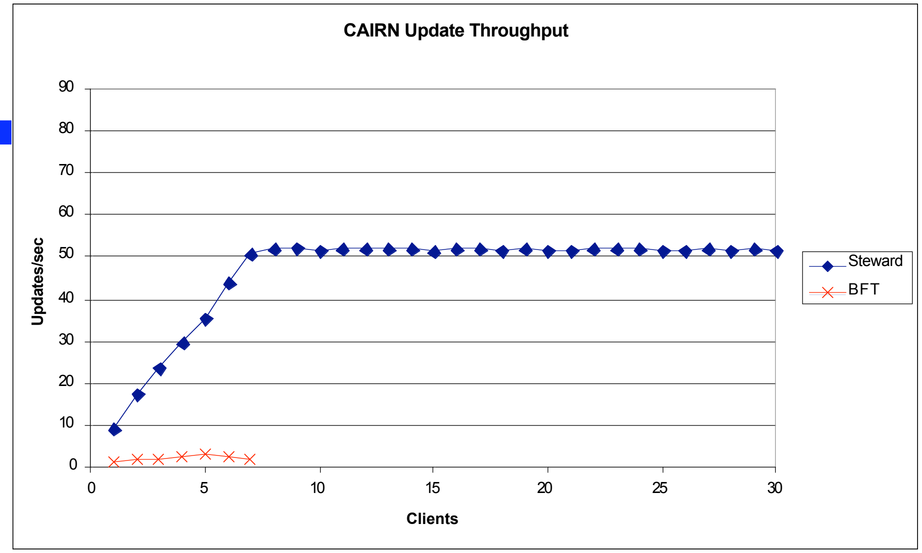
Case 2: Practical Wide-Area Network

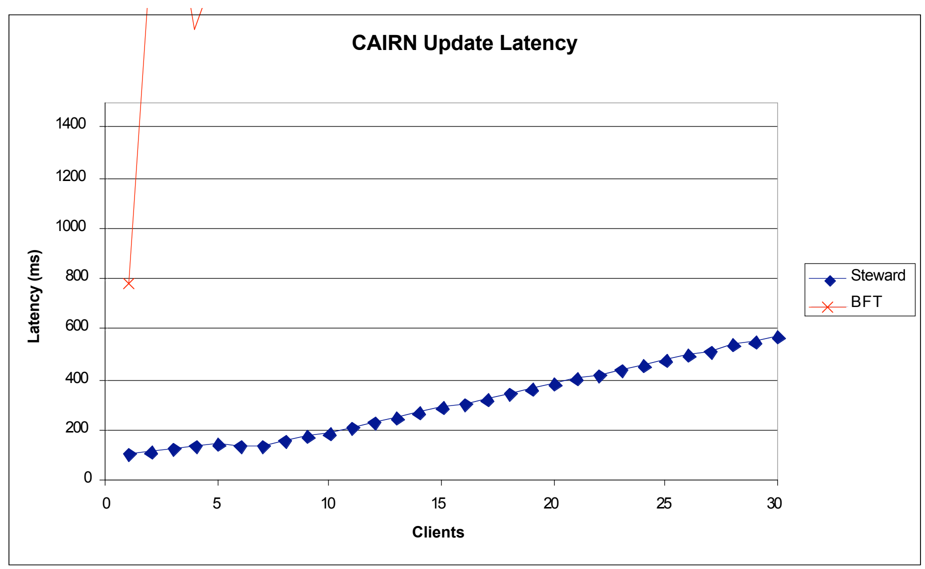


- Based on a real experimental network (CAIRN).
- Modeled on our cluster, emulating bandwidth and latency constraints, both for Steward and BFT.

CAIRN Emulation

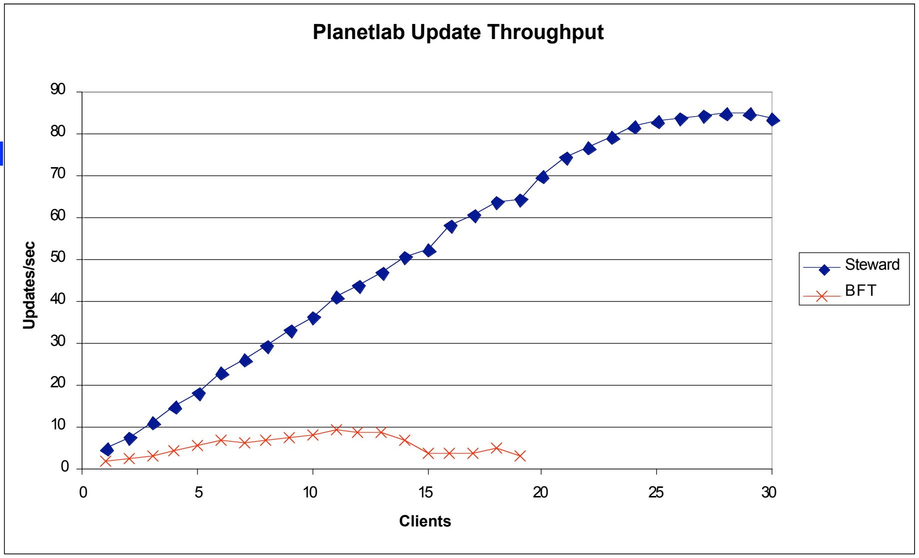
- Link of 1.86Mbps between East and West coasts is the bottleneck
- Steward is limited by bandwidth at 51 updates per second.
- 1.8Mbps can barely accommodate 2 updates per second for BFT.
- Earlier experimentation with benign fault 2-phase commit protocols achieved up to 76 updates per second.

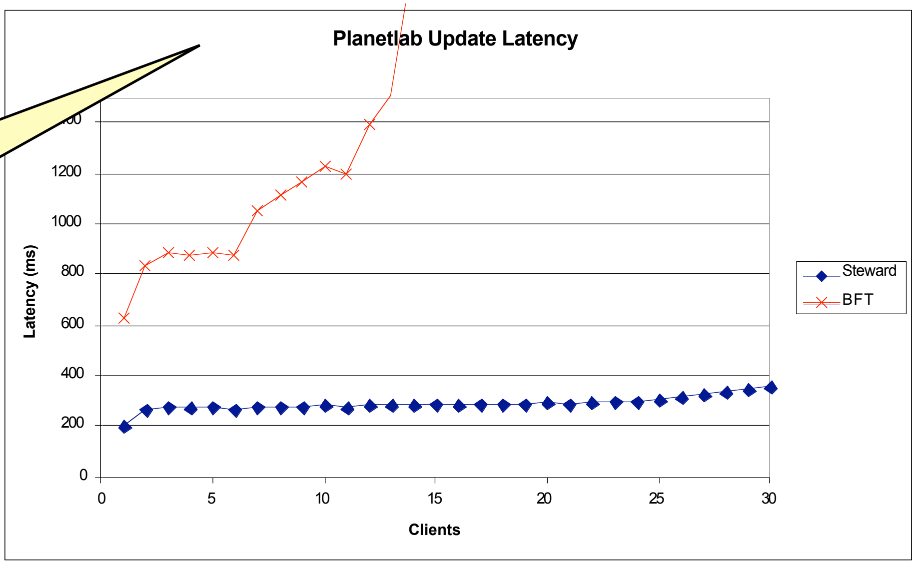




Case 3: PlanetLab

- 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, both for Steward and BFT.





3-fold latency improvement even when bandwidth is not limited.

Performance Summary

- The system can withstand *f* (5) faults in each site.
- Performs better than a flat solution that withstands *f* (5) faults total.
- **Performance**
  - Between twice and over 30 times lower latency, depending on network topology and update/query mix.
  - Program metric met and exceeded in most types of wide area networks, even when write updates only are considered.
- **Availability**
  - Read-only queries can be answered locally even in case of partitions.
  - Write updates can be done when only a majority of sites are connected (as opposed to 2f+1 out of 3f+1 connected servers).

Outline

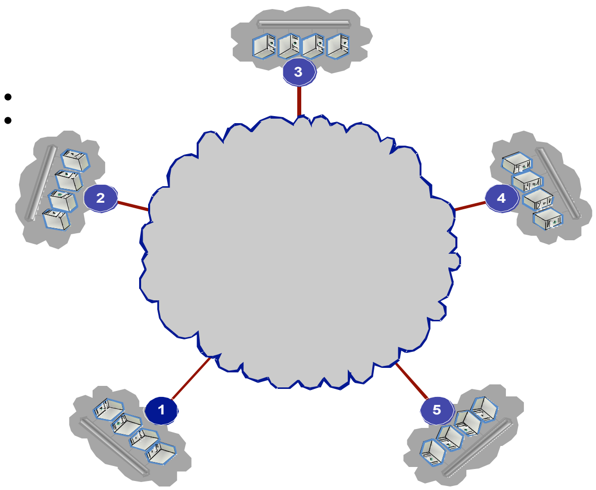
- Introduction
- Overview of state-of-the-art solutions
- Our approach: Steward
- Experimental results
- **Red team experiment**
- Summary

Red Team Experiment

- Performance evaluation – symmetric network
  - Several points on the performance graphs presented were re-evaluated.
    - results were almost identical.
  - Thorough discussions regarding the measuring methodology and presenting the latency results
    - validated our experiments.
  - Five crash faults were induced in the leading site

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.

Summary

- Reduces the message complexity on wide area exchanges from $O(N^2)$ to $O(S^2)$ (N being the total number of replicas in the system, S being the number of wide area sites)
- The improved performance and availability are obtained by containing Byzantine behavior within a site
- Implemented a prototype that passed a red-team experiment

Other Projects Focused on the Insider Threat Model at DS^2 Lab

- **Survivability of ad hoc and hybrid wireless networks**: current focus on position-based or geographical routing
- **Survivable overlay networks**: looked at control attacks in adaptive overlay networks

Contact Information

EMAIL: crisn@cs.purdue.edu

URL:

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

THANK YOU!