

Replication

Amin Vahdat

CSE 123B

May 25, 2006

Annoucements

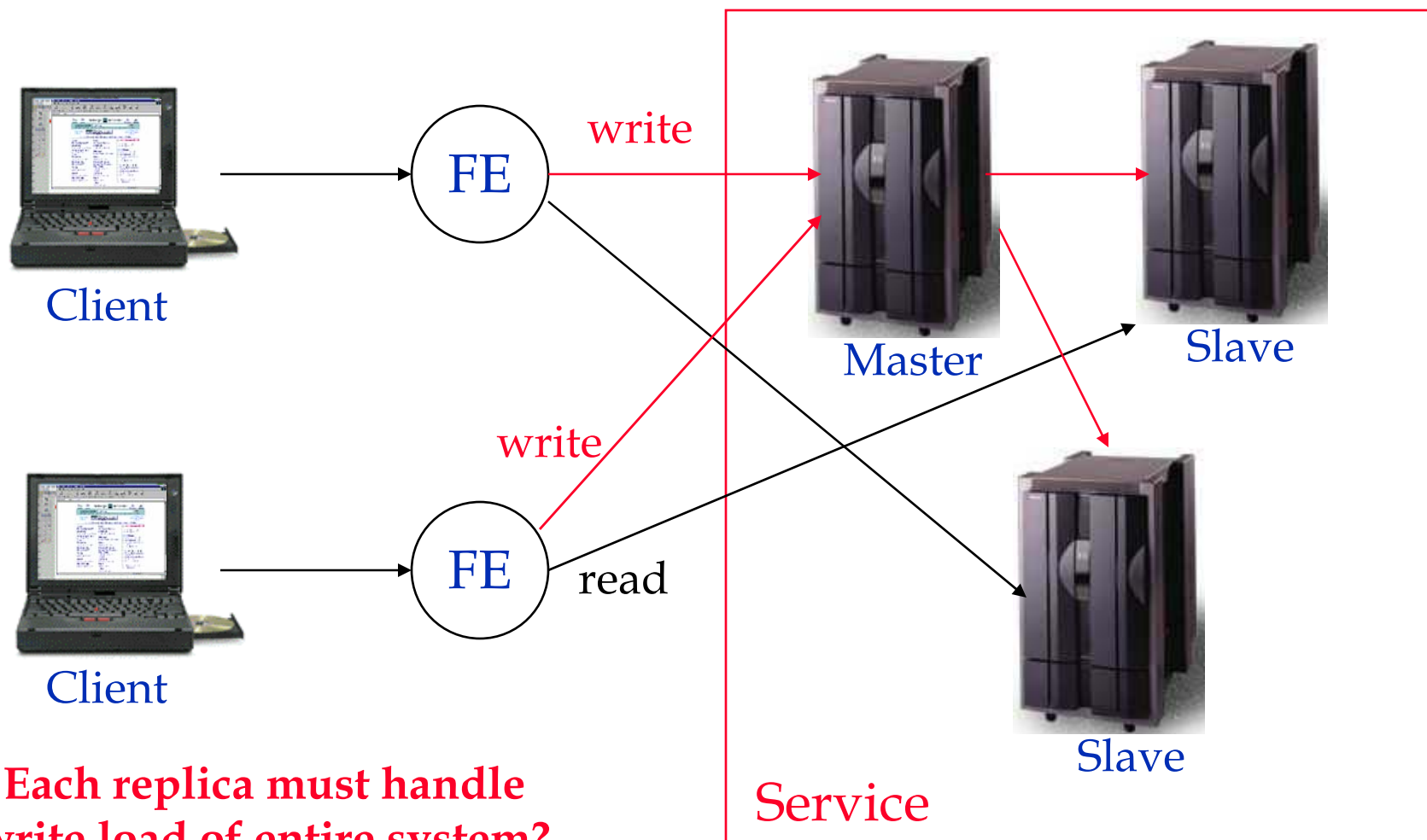
- Third assignment
 - Due date June 9, 5 pm
- Final exam, June 14, 11:30-2:30

Replication

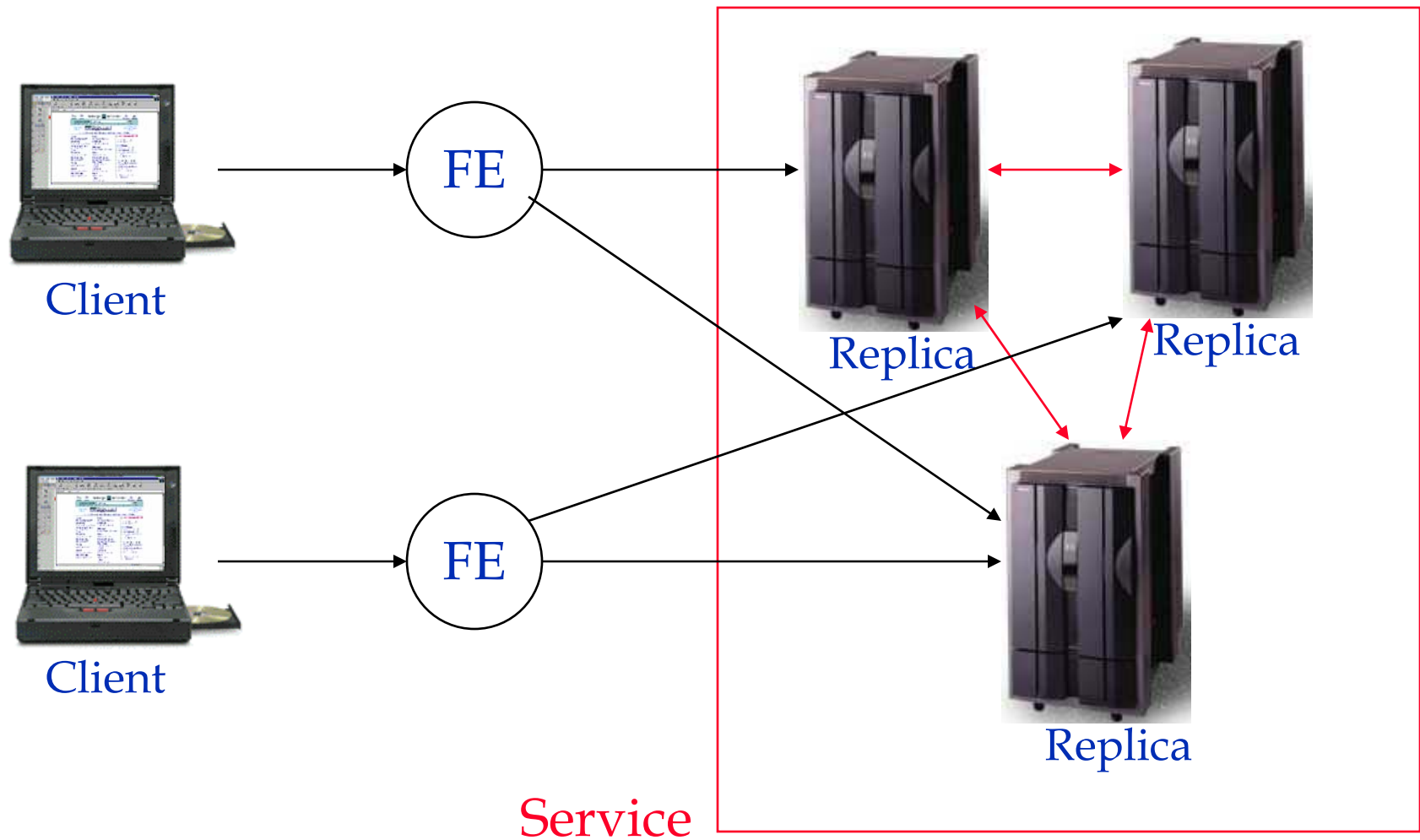
How to Handle Updates?

- Problem: all updates must be distributed to all replicas
 - Different consistency guarantees for different services
 - Synchronous vs. asynchronous update distribution
 - Read/write ratio of workload
- Primary copy
 - All updates go to a single server (master)
 - Master distributes updates to all other replicas (slaves)
- Gossip architecture
 - Updates can go to any replica
 - Each replica responsible for eventually delivering local updates to all other replicas

Primary Copy



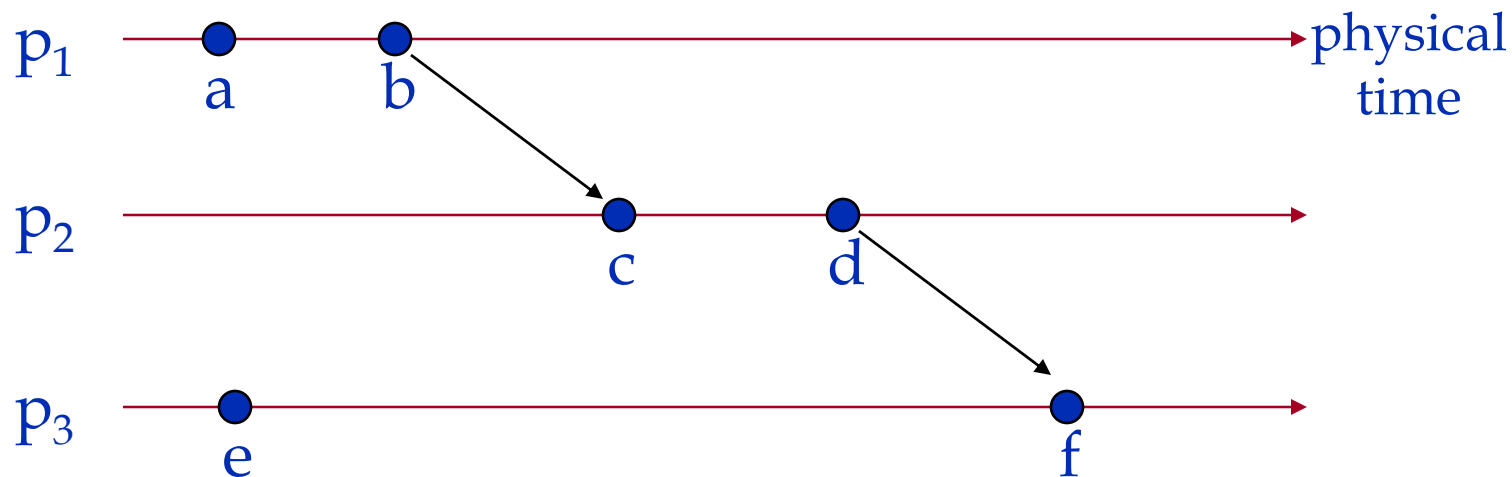
Gossip Architecture



Update Ordering Requirements

- Total Order
 - Bulletin board: all messages assigned globally unique message identifier
 - For messages $r1$, $r2$, either $r1$ appears before $r2$ at all replicas or $r1$ appears after $r2$ at all replicas
- Causal Order
 - Bulletin board: message replies appear *after* original posting
 - For messages $r1$, $r2$, $r1$ appears before $r2$ if $r1$ *happens before* $r2$

Happens Before



- Captures potential causal ordering (information flow)
- $a \rightarrow b$ if a takes place before b in same process
- $\text{Send}(m) \rightarrow \text{recv}(m)$
- Transitivity holds

Implementing Total Ordering

- Use *sequencer*
 - Send updates to centralized site, assign monotonically increasing identifier, distribute to all replicas
 - Single point of failure, contention
- Distributed total ordering
 - Front end sends update to all replicas
 - Each replica proposes unique id
 - Front end picks highest value
 - Transmits final value back to replicas
 - 3 messages/replica overhead

Network Partitions

- Some failure (either network or host) keeps replicas from communicating with one another
- How to proceed with read/write transactions in case where not all replicas can be contacted?

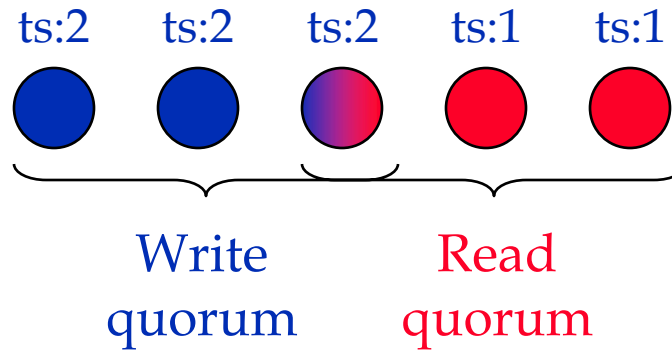
Network Partitions

- Some failure (either network or host) keeps replicas from communicating with one another
- How to proceed with read/write transactions in case where not all replicas can be contacted?
 - Optimistic versus pessimistic techniques
- Optimistic: proceed as normal, resolving conflicts later
 - Coda
- Pessimistic: ensure that replicated database can eventually be restored to consistent state without user intervention
 - Assumes that partition will eventually be repaired
 - What if the network is never fully connected?

Quorum (Voting) Consensus Techniques

- Allow updates to be completed even when only a subset of replicas are available
 - Or have the latest version of a data item
 - May need to bring some set of replicas up to date to proceed
- One technique is the weighted voting scheme
 - Assign a number of votes to each replica
 - Determine a write quorum W and a read quorum R
 $W > \text{half the total votes}$, $R + W > \text{total number of votes}$
 - Ensures that there is some overlap between read and write quorums
Read operation guaranteed access to one site with latest version

Quorum Example



- 5 replicas, read quorum: 3, write quorum: 3
 - $R + W > \#$ votes ensures overlap between any read/write quorum
 - $W > \frac{1}{2}$ votes ensures overlap between two write quorums

Quorum Consensus

- Write operations can be propagated in background to replicas not in quorum
 - Assumes eventual repair of any network partition
- Operations are slowed by the necessity of first gathering a quorum
 - Though previously, all writes had to go to all replicas
 - With quorum system, must only contact subset of replicas

Byzantine Failures

- Normally consider failures to be *fail-stop*
 - Meaning that a process/machine goes down without doing anything incorrect
 - In real world, corrupted memory means that failures are not always fail stop
- Consider worst-case scenario where machines/processes fail and become malicious
 - Can send arbitrary messages
 - Can lie about their identity

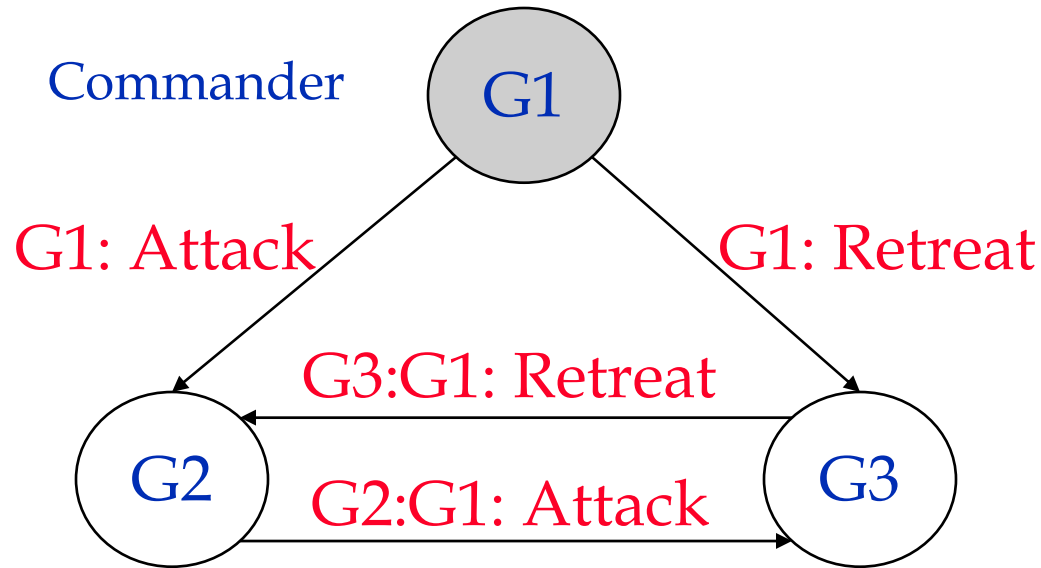
Consensus and Byzantine Failure

- Byzantine Generals problem:
 - N Generals encamped outside of enemy city
 - Would like to come to consensus as to whether they should attack at dawn
 - Use messengers to communicate
 - Assume messages are not lost
 - Can identify sender of each message (but no signatures)
 - What happens when one or more of the generals fails (becomes malicious)?

Byzantine Consensus Goals

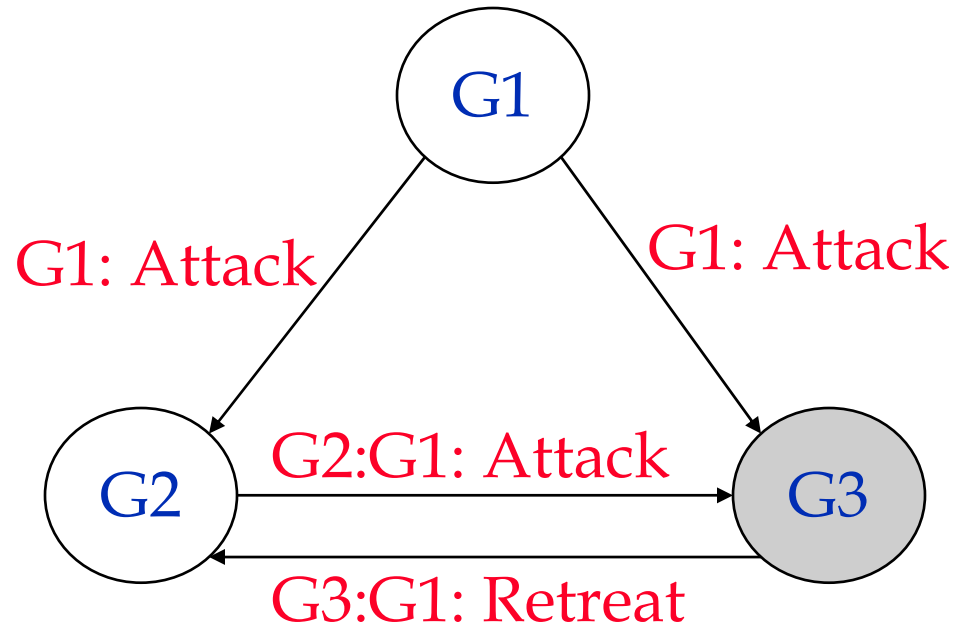
- Termination
 - Eventually, each correct process sets its decision variable
- Agreement
 - The decision value of all correct processes is the same
- Integrity
 - If the commander is correct, then all correct processes decide on the value the commander proposed

Byzantine Consensus



Neither G2 nor G3 can determine that G1 is bad;
something is wrong but attack or retreat?

Byzantine Consensus



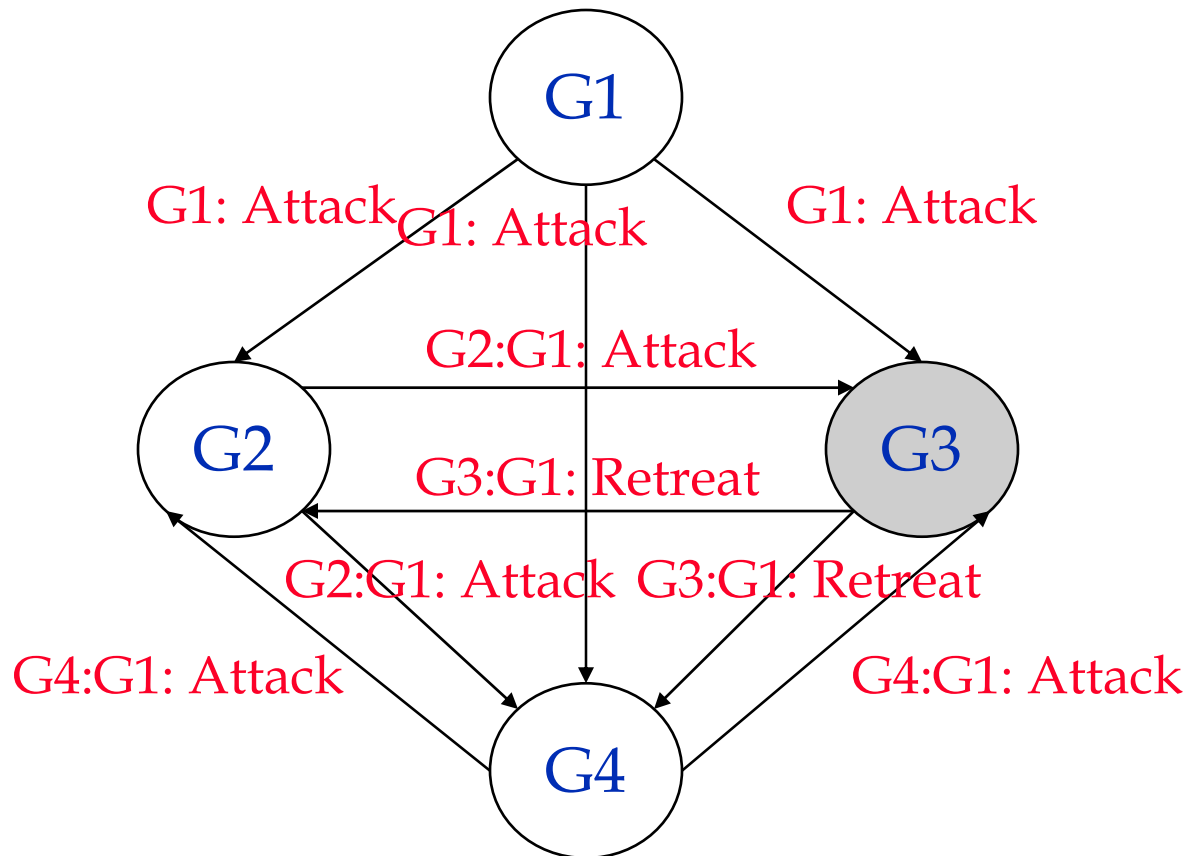
G2 realizes that someone is faulty, but who?

➔ G3 in this case

Byzantine Consensus

- In the absence of authentication:
 - $3m+1$ servers required to detect the presence of m traitors
- Consider mission critical system trying to reach consensus
 - N replicated servers performing the same calculation
 - Each server votes for the decision
 - Majority wins
 - Assume “good” (non-faulty) servers all come to the same conclusion

Byzantine Generals



- 4 processes,
1 faulty

A System and Language for Building System Specific, Static Analyses