

Towards a practical byzantine fault-tolerant stack

V. Arun

UMass Amherst Computer Science

NSF Wireless Security Workshop

Atlanta, April 10 2008

The network as a distributed system

- Goals: The network should
 - provide a route between two good nodes unless partitioned by faulty ones
 - provide a fair share of network resources to flows (src-dst pairs)
- Threats: A fraction of byzantine nodes may behave arbitrarily
 - drop or delay packets
 - generate useless packets (DoS)

The network as a distributed system

- Goals: The network should
 - provide a route between two good nodes unless partitioned by faulty ones
 - provide a fair share of network resources to flows (src-dst pairs)
- Threats: A fraction of byzantine nodes may behave arbitrarily
 - drop or delay packets
 - generate useless packets (DoS)
- Assumptions: byzantine nodes are computationally bounded
- Mechanism to support second goal in a single contention domain exists

Why difficult?

- Consider a network with total of N nodes with $B < N$ byzantine
 - How does route unavailability depend on B , N , and current network topology?
 - How does flow throughput degrade with B , N , and current network topology?
- Today, even in a benign wireless network environment
 - Route unavailability, oscillations, loops common
 - Some multi-hop TCP flows get completely shut out

Why difficult?

- Consider a network with total of N nodes with $B < N$ byzantine
 - How does route unavailability depend on B , N , and current network topology?
 - How does flow throughput degrade with B , N , and current network topology?
- Today, even in a benign wireless network environment
 - Route unavailability, oscillations, loops common
 - Some multi-hop TCP flows get completely shut out

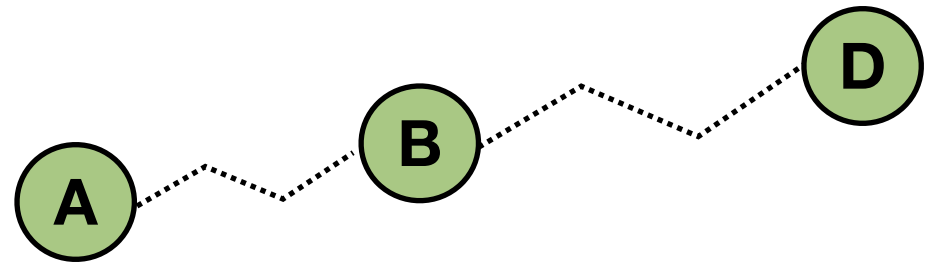
Architectural problems a legacy of wired network stack

C1: Implementing routing security policies

- Common routing protocols
variants of shortest path (LS,
DV, PV)
 - detecting or recovering from
misbehavior complex ([Hu/
Perrig], [Awerbuch], [Haas])
- Need mechanism for security
policy driven routing
 - convergence should be
verifiable, not assumed

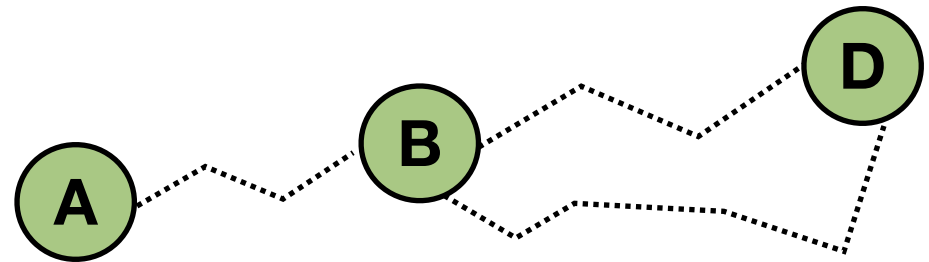
C1: Implementing routing security policies

- Common routing protocols variants of shortest path (LS, DV, PV)
 - detecting or recovering from misbehavior complex ([Hu/Perrig], [Awerbuch], [Haas])
- Need mechanism for security policy driven routing
 - convergence should be verifiable, not assumed



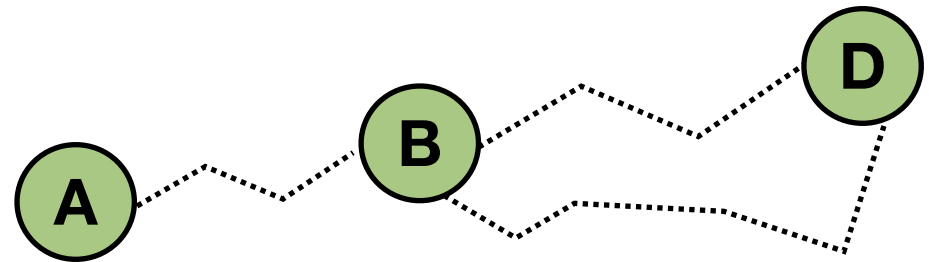
C1: Implementing routing security policies

- Common routing protocols variants of shortest path (LS, DV, PV)
 - detecting or recovering from misbehavior complex ([Hu/Perrig], [Awerbuch], [Haas])
- Need mechanism for security policy driven routing
 - convergence should be verifiable, not assumed



C1: Implementing routing security policies

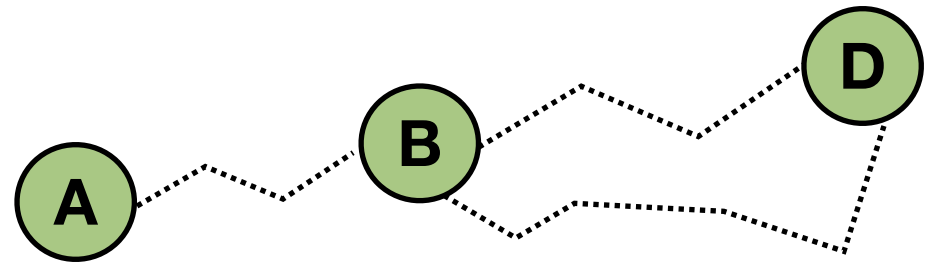
- Common routing protocols variants of shortest path (LS, DV, PV)
 - detecting or recovering from misbehavior complex ([Hu/Perrig], [Awerbuch], [Haas])
- Need mechanism for security policy driven routing
 - convergence should be verifiable, not assumed



Consistency: If A's adopted route is A...B...D, then B's adopted route is B...D

C1: Implementing routing security policies

- Common routing protocols variants of shortest path (LS, DV, PV)
 - detecting or recovering from misbehavior complex ([Hu/Perrig], [Awerbuch], [Haas])
- Need mechanism for security policy driven routing
 - convergence should be verifiable, not assumed

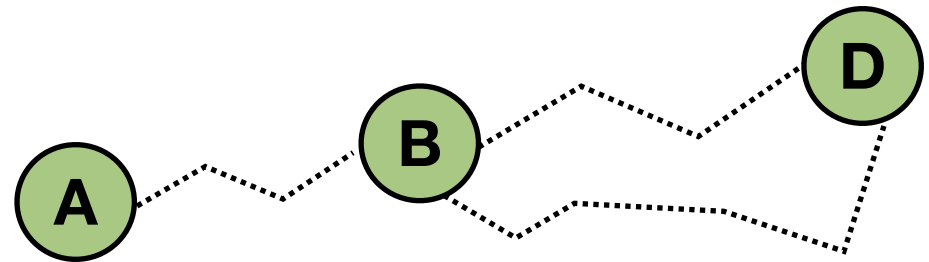


Consistency: If A's adopted route is A...B...D, then B's adopted route is B...D

- Consensus routing: Use a route only after all dependent nodes have verifiably agreed to the route

C1: Implementing routing security policies

- Common routing protocols variants of shortest path (LS, DV, PV)
 - detecting or recovering from misbehavior complex ([Hu/Perrig], [Awerbuch], [Haas])
- Need mechanism for security policy driven routing
 - convergence should be verifiable, not assumed



Consistency: If A's adopted route is A...B...D, then B's adopted route is B...D

- Consensus routing: Use a route only after all dependent nodes have verifiably agreed to the route
- Multipath routing: End-host rate control for highly responsive forwarding security

C2: Enforceable fair resource allocation

- How to allocate network resources in a byzantine environment?
 - network capabilities for verifiable resource allocation

C2: Enforceable fair resource allocation

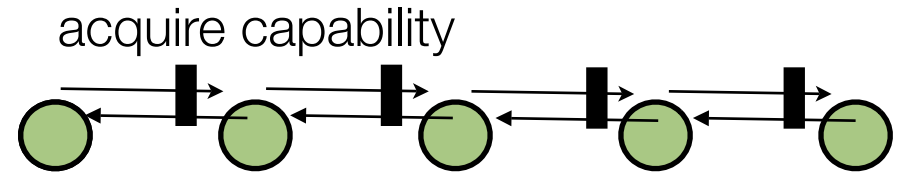
- How to allocate network resources in a byzantine environment?



- network capabilities for verifiable resource allocation

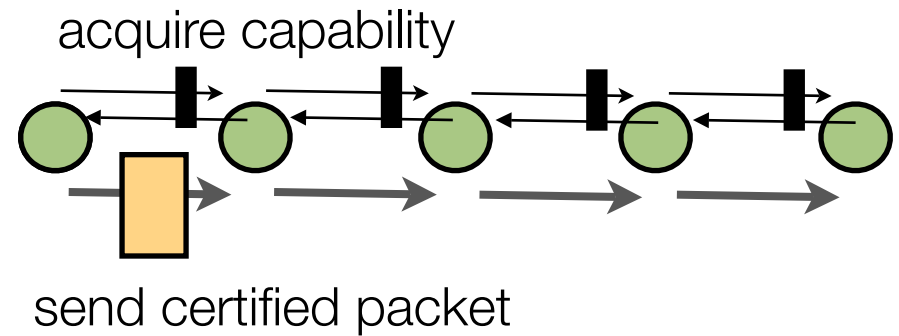
C2: Enforceable fair resource allocation

- How to allocate network resources in a byzantine environment?
- network capabilities for verifiable resource allocation



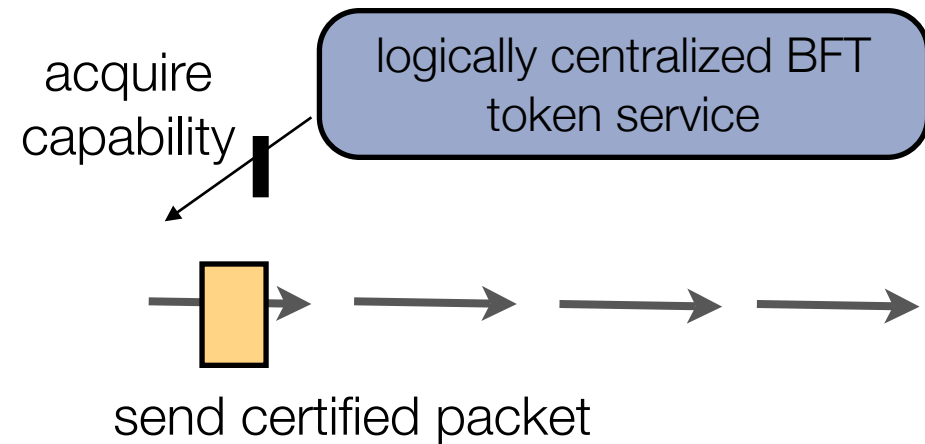
C2: Enforceable fair resource allocation

- How to allocate network resources in a byzantine environment?
 - network capabilities for verifiable resource allocation



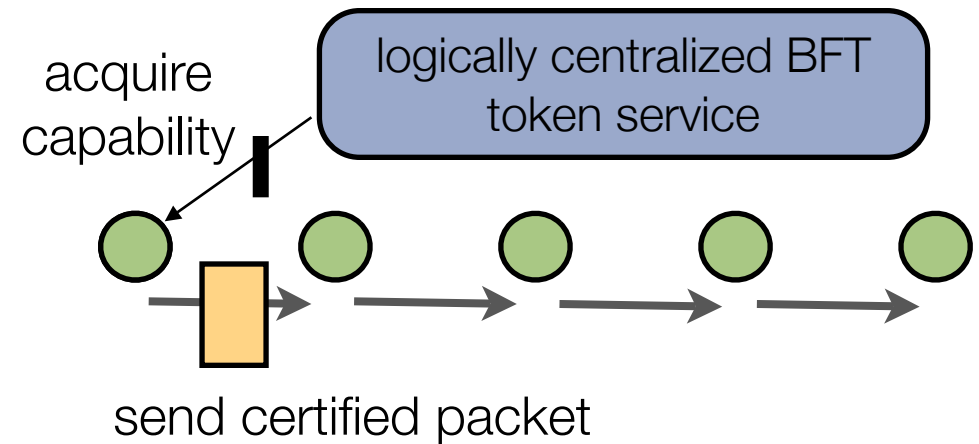
C3: Monitoring and accountability

- Byzantine fault-tolerant distributed service to account for resource usage
- leverage traditional BFT replicated state machine or quorum techniques



C3: Monitoring and accountability

- Byzantine fault-tolerant distributed service to account for resource usage
- leverage traditional BFT replicated state machine or quorum techniques



C4: Gracefully handling persistent partitions

- Protocol stack today assumes contemporaneous e2e path
 - OLSR, AODV, DSR
 - TCP, UDP
- Breaks down in disruption-prone environments

C4: Gracefully handling persistent partitions

- Protocol stack today assumes contemporaneous e2e path
 - OLSR, AODV, DSR
 - TCP, UDP
- Breaks down in disruption-prone environments
- DTN routing
 - replication more robust than forwarding only
 - treats routing+transport as a resource allocation problem
 - very different from OLSR and TCP!

C4: Gracefully handling persistent partitions

- Protocol stack today assumes contemporaneous e2e path
 - OLSR, AODV, DSR
 - TCP, UDP
- Breaks down in disruption-prone environments
- DTN routing
 - replication more robust than forwarding only
 - treats routing+transport as a resource allocation problem
 - very different from OLSR and TCP!

Challenge: How to gracefully degrade performance from well-connected mesh backhauled to *always-partitioned* mobile environments with byzantine nodes?

Conclusions

- Problem

- Every layer in the protocol stack vulnerable to presence of even a single byzantine node

- Design process

- Treat protocol at each layer as a distributed computation
- Design protocol to tolerate a fraction of byzantine nodes