Emergence of new mechanisms for Transparent Bridges

ARP-Path as a New Bridging Mechanism

Unrestricted flood search bridging, diversified path, non deterministic bridging

Guillermo Ibanez GISTNetserv group. Universidad de Alcalá.Madrid. Spain.
Supported by MEDIANET and EMARECE projects
IEEE Plenary Meeting 13-18 March 2011. Singapore.

Universidad de Alcalá
Dpto. de Automática
Área de Ingeniería Telemática

UNION EUROPEA
FONDO SOCIAL EUROPEO

La Suma de Todos
Consejería de Educación
Comunidad de Madrid
www.madrid.org

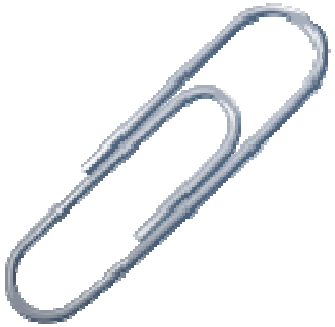
Junta de Comunidades de
Castilla-La Mancha

Contents

- Introduction.
- Broad-Path mechanism (base of the ARP-Path protocol)
  - Basic operation
  - Implementations
  - Simulation results:
    - Throughput and load distribution
    - Compatibility with standard bridges
  - Applicability to IEEE 802.1 protocols
  - The road ahead
- Conclusion

Research statement

- Protocols tend to complexity in their way to standardization
- Our basic challenge:
  - Is it possible a shortest path/low latency protocol based solely on *bridging mechanisms*?
  - Is it possible to keep it as *simple* as bridges are?
    - lower cost, higher performance
  - Just to provide more room for the unavoidable added complexity when standardizing



Update from Volterra Interim 9/2009

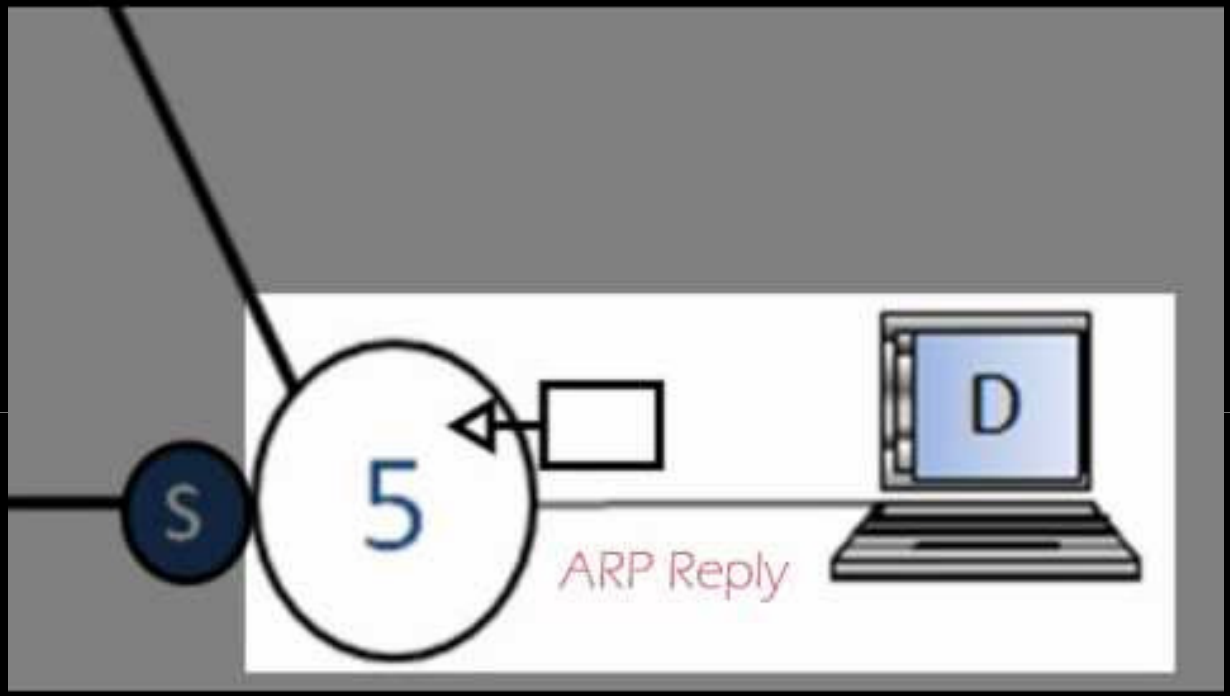
- FastpathUD was presented at Volterra Interim Meeting 9/2009
  - Used Up/Down turn prohibition protocol to prevent loops (and required RSTP as ancillary for that)
  - Fortunately, neither Up/Down nor RSTP are needed to prevent loops.
  - Renamed to ARP-Path protocol and Broad-Path as the generic mechanism
- ARP-Path has been greatly simplified!

Essentials of Broad-path (ARP-Path)

- A new layer two flood search&learn mechanism is proposed that:
  - Finds low latency unicast path between hosts
  - “ “ “ “ “ if used between bridges
  - **Sets up instantly trees** rooted at source host or bridge
  - Distributes multicast and broadcast frames without loops
  - **Distributes load** among available routes, based on path latency.
  - *See back up slides for a visualization of the mechanism*

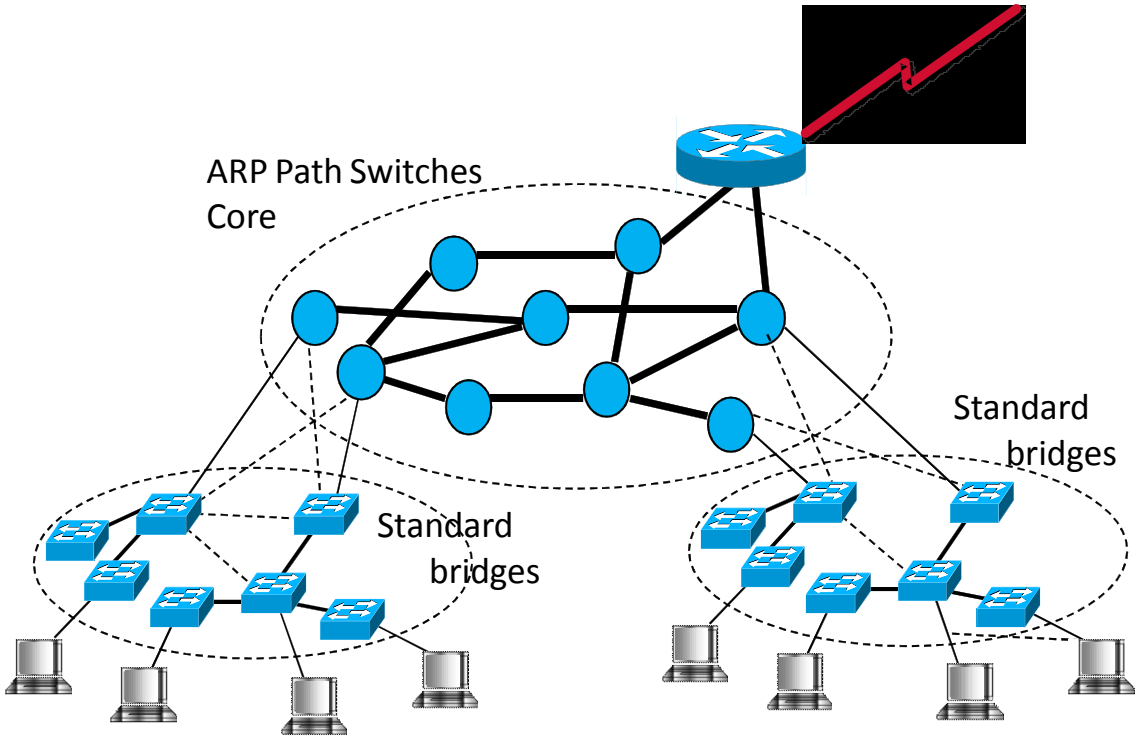
Transparent bridges with low latency paths.

- Minimizing protocol messages:
  - *Reuse of the standard broadcast ARP messages to set up paths and temporary trees*
    - ***No extra message cost*** *to set up paths (excluding replication at redundant links)*
- How to avoid broadcast loops?*:*
  - ***Limit source address learning: locking the learning of the address*** *to the port of the bridge that receives first the frame. Learning occurs only with ARP messages.*
  - ***Discard for a short time all broadcast frames received via a different port***


S
5
D
ARP Reply

Loop prevention

ARP-Path bridges are compatible with standard bridges in core-island mode



- ARP Path bridges become root of spanning trees of standard bridges (announce a high priority virtual root bridge)
- Islands split in two or more trees if connected with redundant links to core
- Auto configuration as ARP-Path core is possible via protocol migration mechanisms (STD bridges are forced to core periphery)

ARP Path complexity

- Stored state is similar to transparent bridges: same number of MACs to learn per port, two persistence timers to process (lock-short, learn-long). Spanning tree protocols not required.
  - Extra packets received on ports not associated to source address must be discarded. Suitable for CAM-based hardware implementations or new ones.
- Reconfiguration and network availability:
  - Only the affected paths being used require path repair.
  - Path diversity (per-host, on-the-fly paths) provides robustness and high network availability.
  - Full MAC address flush at network (like RSTP) is also possible via ARP Path TCNs.

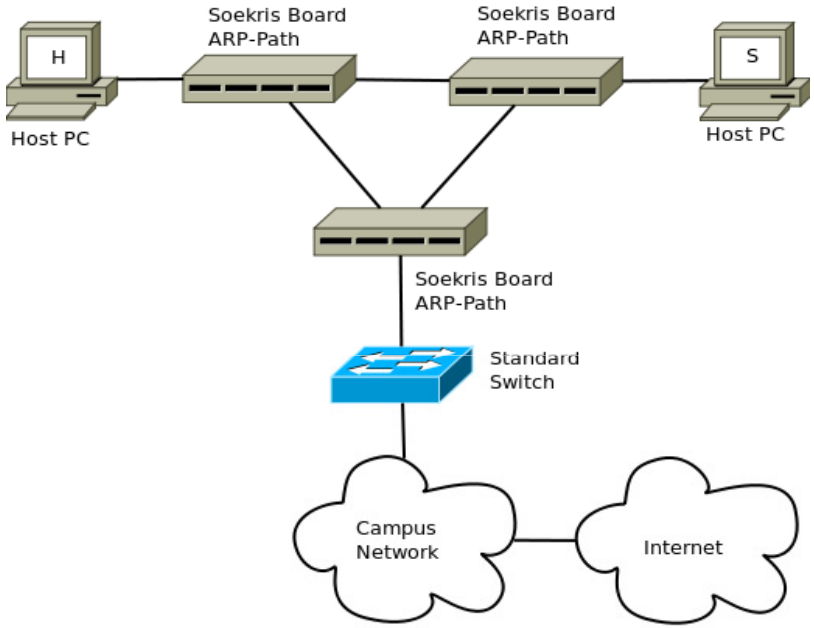
ARP Path: broadcast

- Extra flooded packets on redundant links: small percentage of the total of links (the highest fraction of links are the non redundant host-switch links).
- Reducing ARP messages: implement ARP Proxy function (like Etherproxy [4]) on ARP Path bridges.
  - Proxy implementation requires basically to add an IP field to the bridge table.
- Frequently used addresses (active servers) remain in ARP proxy caché.

Implementations and demos

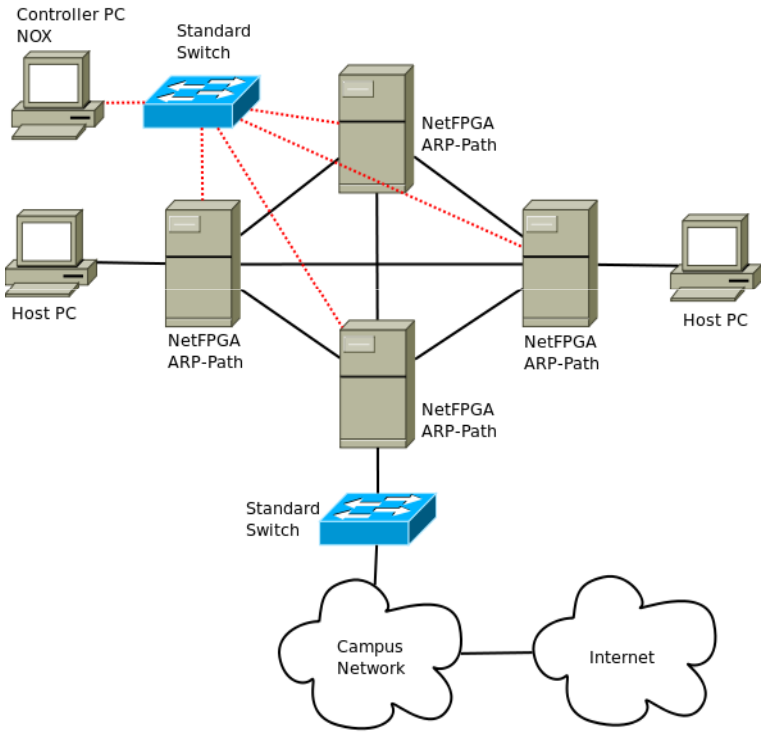
- Completed (best demo award at LCN 2010) :
  - Openflow/NetFPGA
  - Linux 2.6 (Soekris boards)
- Ongoing:
  - Linksys WRT160N (DD-WRT)
    - For bigger test networks
  - NEC Openflow Switch
  - Hardware: NetFPGA, other…
- Videos of demos available at : http://wn.com/gistnetserv

First implementation (Linux) [2]



- On kernel and user space using *ebtables* [5]
- Functionally simple to code and implement
- All services of campus network operate smoothly (DHCP, video streaming)
- Delays similar to hardware switches (on kernel part)

Second implementation (Openflow/NetFPGA) [2]



- 4 NetFPGA with 4*1 Gbps links
- ARP-Path protocol logic resides at NOX controller (as flow rules to ARP-Path switches)
- Functionally simple to code, implement and modify
- All services of campus network operate smoothly (DHCP, video streaming)
- Delays similar to hardware switches in normal forwarding .
- Robust and fast reconfiguration after link failure.

Omnet. Throughput of simulated paneuropean reference network [1]

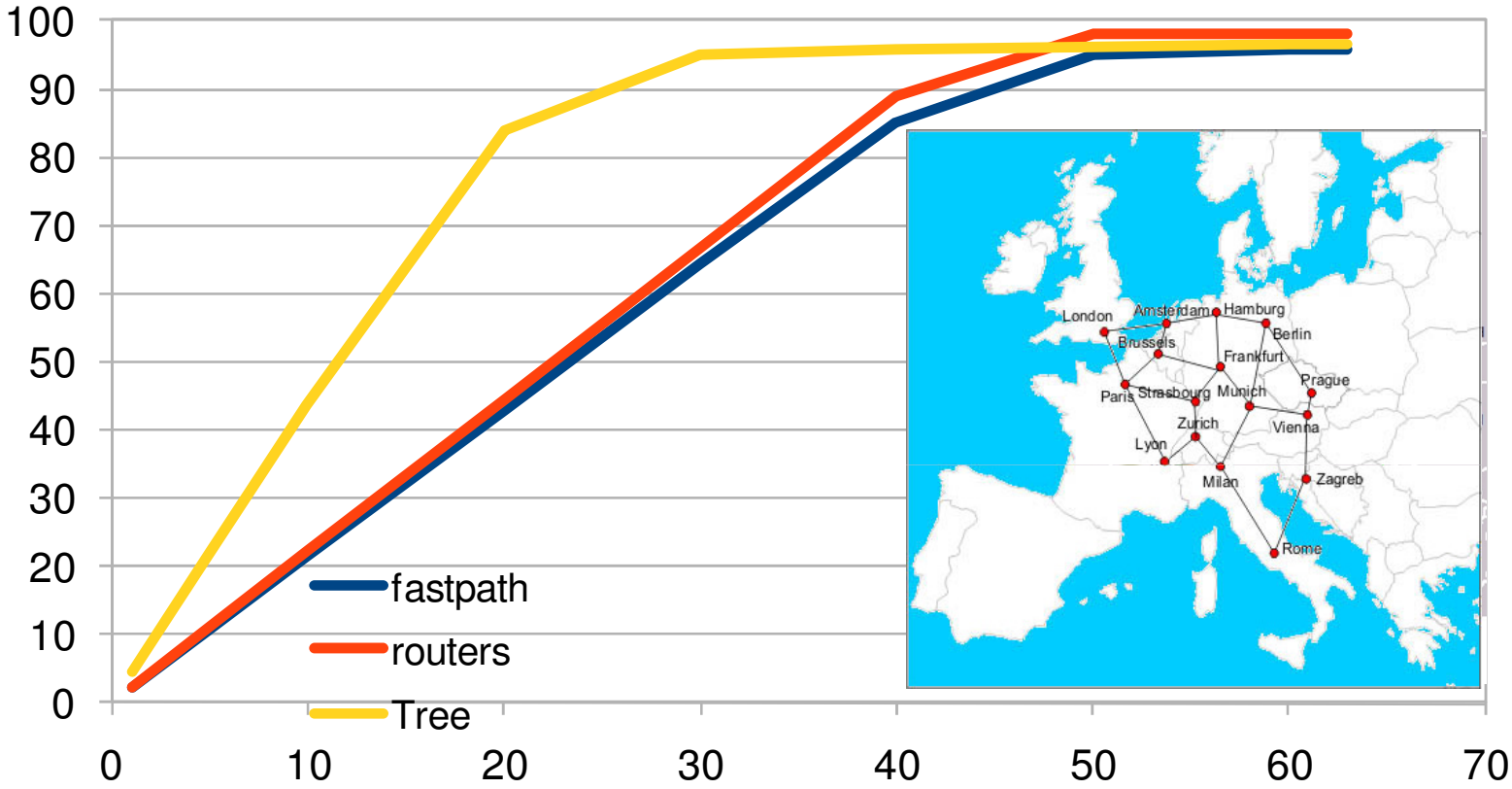
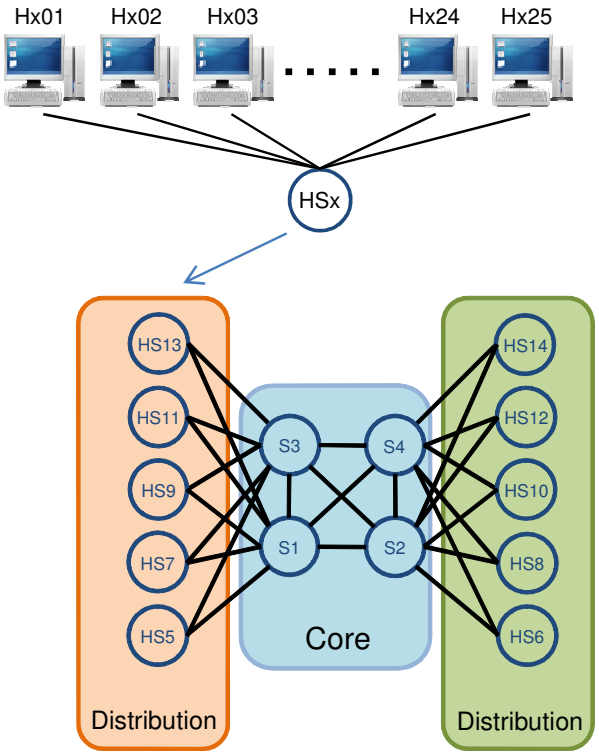
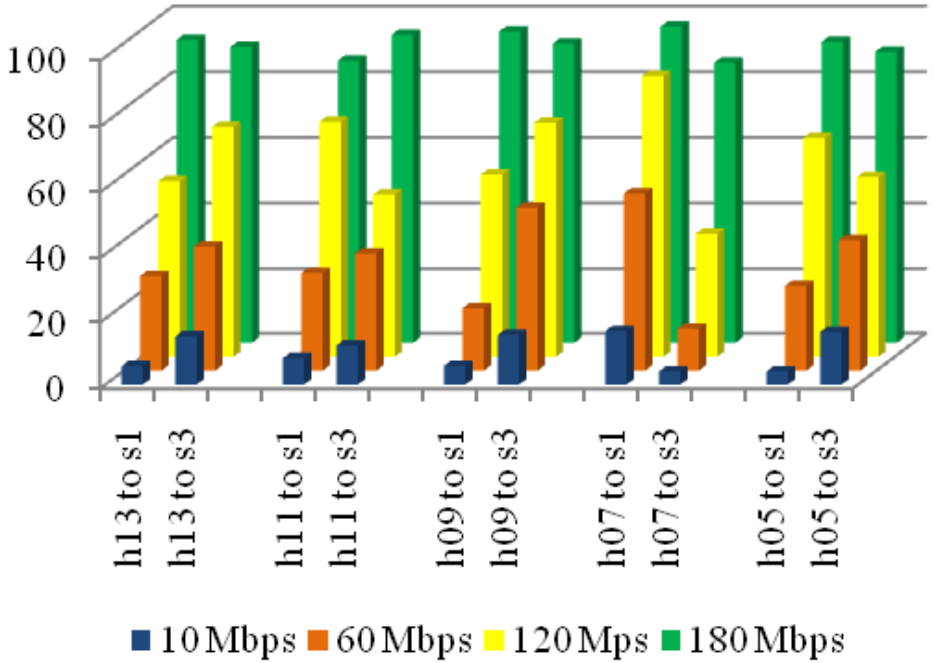


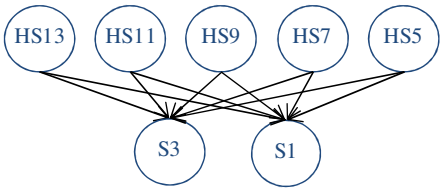
Figure 9. Throughput comparison of pan european network in % of most loaded link versus % of average traffic load applied at the sending host link [1]

Automatic load split between redundant paths of data center [6]

- Two level data center topology, 25*10 hosts
- UDP traffic from hosts on the left to hosts on the right
- Increasing load at hosts to reach link saturation
- Load is distributed among the pairs of links between distribution switch (hs13,hs11,hs9,hs5) and core (s1,s3)







Predictability, controllability, manageability

- ARP-Path is *inherently effective:* it finds, with zero added latency, the *best available path at the time it is needed*.
- ARP-Path does not provide a predictable, deterministic path
  - Is predictability of path essential when reliability and performance are high ?.
    - ARP-Path flooding itself provides high path availability
  - Controllability and predictability may be added (bridges/ports not allowed to execute ARP-Path, etc)
    - V.g. : fine tune and configure at bridges the priority handling of ARP-Request frames (latency, queues, forwarding preferences).
  - ARP-Path is somehow an “autonomic” protocol: it finds paths and balances load according to the latencies.
    - *Autonomic protocols need autonomy*.

Predictability, controllability, manageability

- Manageability: Bridges and ports can be included or excluded from the ARP Path protocol via SNMP.
- Can coexist with spanning tree or SPB protocols (separation by VLANs)
- Compatible with standard Connectivity Fault Management mechanisms.

Applicability to IEEE 802.1 protocols

- **802.1D** : add ARP-Path forwarding as a complementary optional protocol to spanning tree.
  - Obtaining simple, efficient low latency switches
- SPBV: add Broad-Path as optional mechanism to set up trees. Interesting as an alternative mechanism.
- SPBM: Broad-Path requires MAC learning with ARP frames.

The road ahead…

(some wishes to move this forward)

- Switch chipset manufacturers:
  - Implement in hardware mechanism for locking address to port and frame discard
  - Develop further suitable hardware mechanisms
- Switch manufacturers: experiment with prototypes, compatibility, new features…
- IEEE 802.1: include as an optional protocol candidate for 802.1D, SPBV, SPBM,…
- **Start considering as an addition to 802.1D (RSTP) seems suited in terms of std work versus performance results.**

Conclusion (ARP-Path pros)

- A new layer two flood-search-learn mechanism is proposed that:
  - Finds **low latency** unicast path between hosts
  - “ “ “ “ “ between bridges
  - **Sets up instantly trees** rooted at source host or bridge
  - Distributes multicast and broadcast frames without loops
  - **Distributes load** among available routes, based on path latency. Path diversity.
- Proven by implementations, next step should be working hardware prototypes.

Conclusion (cons)

- Message overhead consists of extra broadcast replicas at redundant links that are automatically discarded by receiving ports. Redundant links represent a low percentage of total network links.
- Reduction of ARP path broadcast to hosts requires (as other proposals), an ARP Proxy function or centralized or distributed (DHT) host resolution.
  - ARP proxying adds little complexity to ARP-Path switches (only IP info). May increment table size.
- Requires point to point links between ARP-Path switches, as many other advanced protocols.
- Undeterministic paths (the fastest available)

References

[1] Fast Path Ethernet Switching: On-demand, Efficient Transparent Bridges for Data Center and Campus Networks. Guillermo Ibanez, Juan A. Carral, Alberto García-Martínez, José M. Arco, Diego Rivera Pinto, Arturo Azcorra. IEEE LANMAN Workshop. May 2010. http://dspace.uah.es/jspui/bitstream/10017/6298/7/FastpathLANMANcamerareadyv5final.pdf

[2] A Simple, Zero Configuration Low Latency Protocol . Guillermo Ibáñez, Jad Naous, Elisa Rojas, Diego Rivera, Juan A. Carral, José M. Arco. Demo at Conference on Local Computer Networks. October 2010. *Best demo award*. http://dspace.uah.es/jspui/handle/10017/6770

[3]Fast Path bridges: Old and new ideas for the evolution of transparent bridges. (FYI) IEEE 802.1 Interim Meeting Sept 2009. *Primitive protocol proposal that used Up/Down.*

[4] EtherProxy: Scaling Ethernet By Suppressing Broadcast Traffic. Khaled Elmeleegy, Alan L. Cox INFOCOM 2009: 1584-1592

[5] Ebtables. http://ebtables.sourceforge.net

[6] Simulation results of ARP-Path load distribution. http://hdl.handle.net/10017/7829

[7] **Videos of demos** available at : http://wn.com/gistnetserv

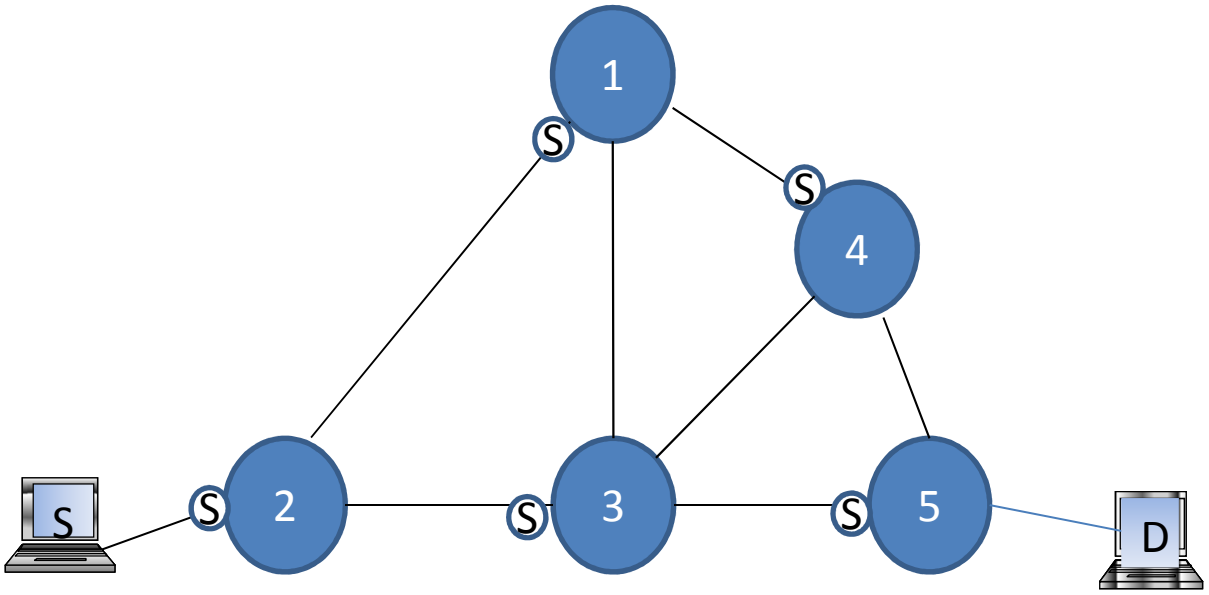
- Thanks for your attention
- Questions?
- Feedback wanted

Back-up slides

	Link State (SPB)	ARP Path
Forwarding state (CAM)	*O(b+h)*	*O(h)*
Routing state	*O(b*d+h)*	*≤ O(h)*
Number of messages	*O(b*E)*	Standard ARP messages + extra flood: *h*(E-N+1)*
Computational complexity	*O(b*log(b) +h)*	One CAM look-up (MAC, port)
Convergence time	*O(path length bs)*	*Negligible (extra processing of ARP at ARP Path bridges)*
Fault recovery	Messages *O(2*E)* Time *O(path length bs)* Recompute *O(b*log(b))*	Messages *O(2*E*hostlink)* Time *O(path length bs)* Recompute *O(b)*
Path diversity computation	*O(b*b*log(b))*	*O(b)*

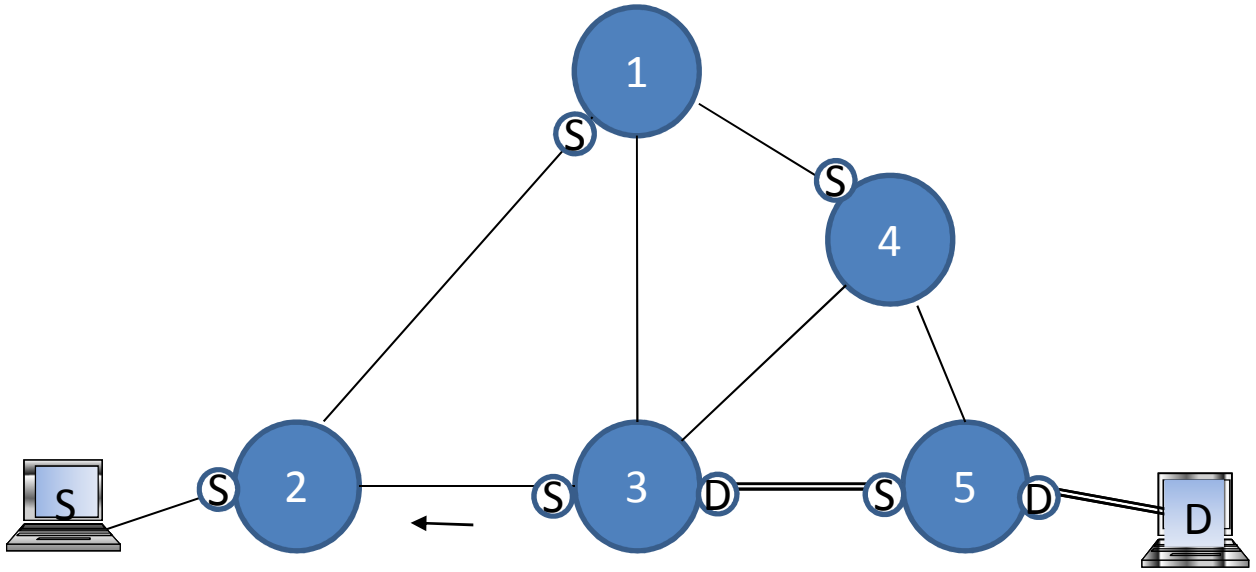
ARP Path basics

- Establish unicast paths and multicast trees just by controlled flooding of a broadcast frame e.g.: ARP Request.
- A temporary tree is established towards the source by ***learning and locking the source address to the port of the bridge that receives first the frame.***



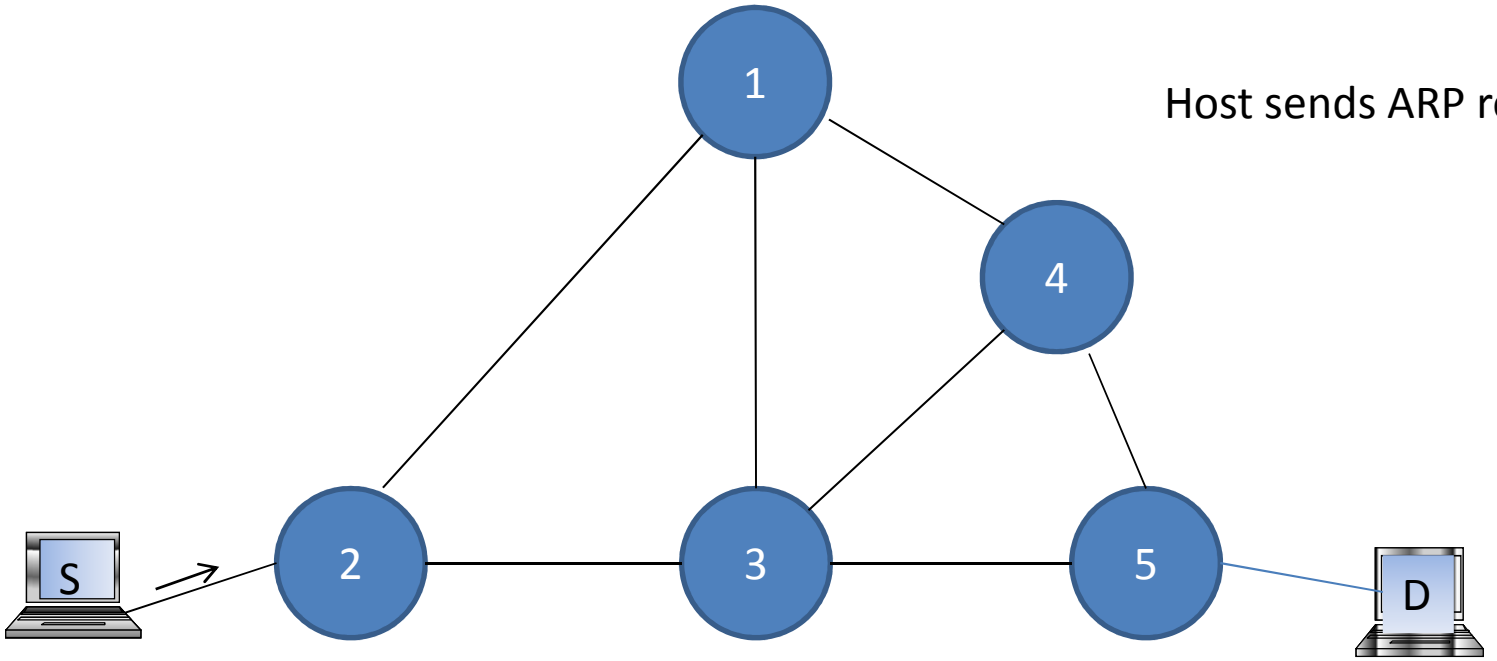
ARP Path basics

- The path in the opposite direction (to destination host) is created by the unicast ARP Reply frame traversing the network from destination host towards source and is the symmetric path of the ARP Request path.



Path set up from host S 1

ARP Request

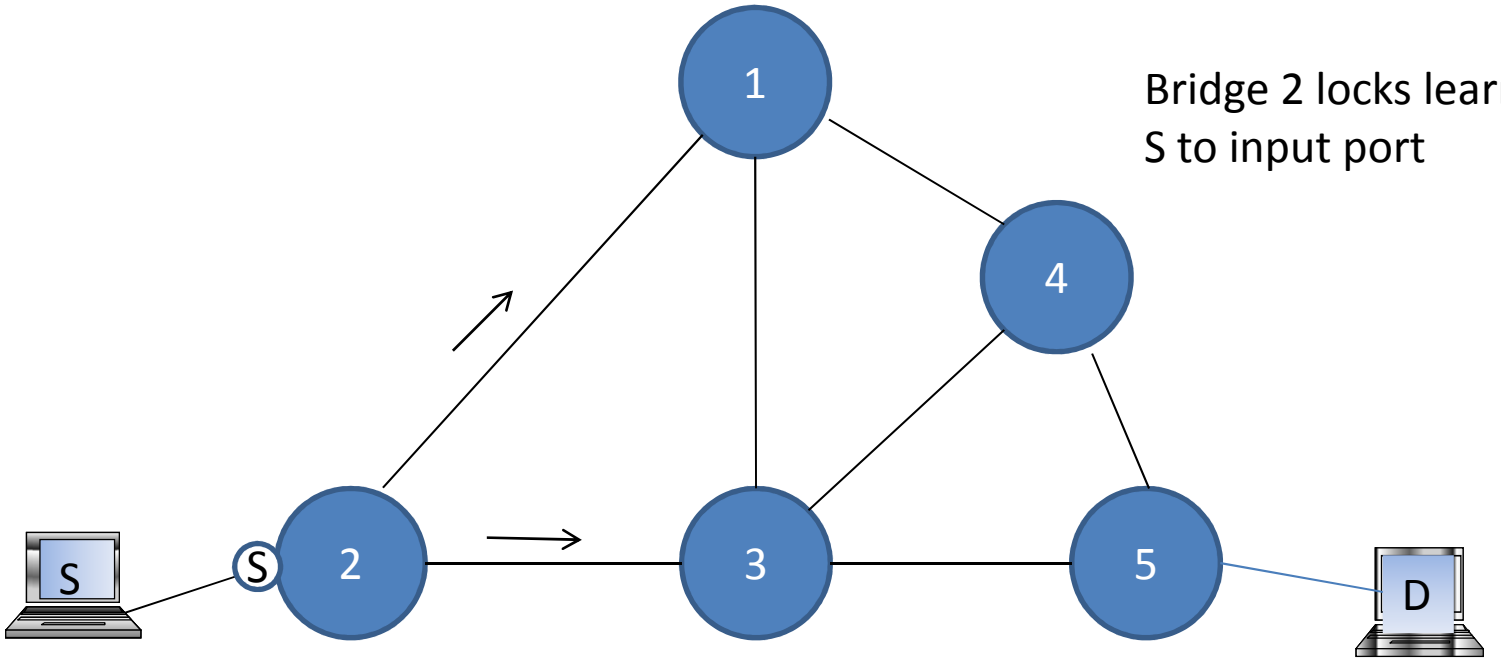


Host sends ARP request frame

ARP Request (broadcast)

Path set up 2

ARP is flooded



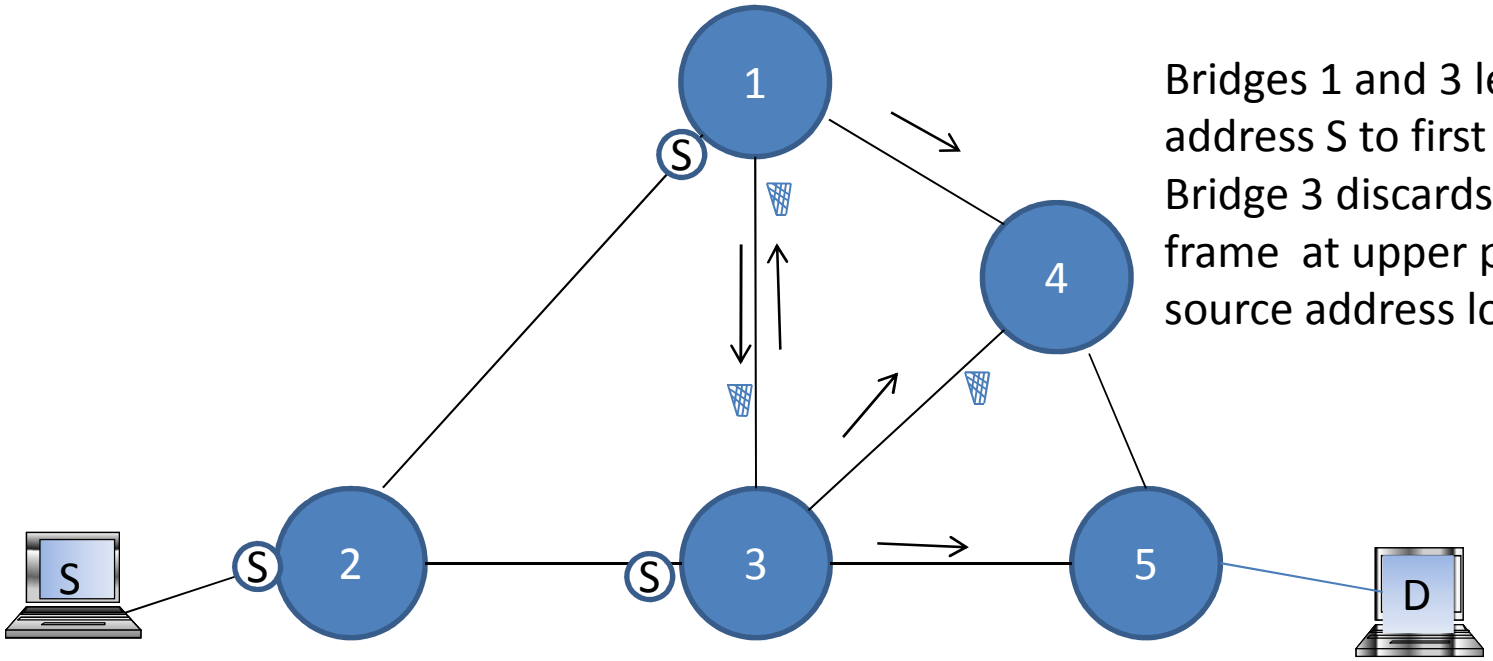
Bridge 2 locks learning of address S to input port

S *Port locked to S*

ARP Request (broadcast)

Path set up 3

ARP propagates through all links



Bridges 1 and 3 learn and lock address S to first input port (left)
Bridge 3 discards late (duplicate) frame at upper port because source address locked

S *Port locked to S*

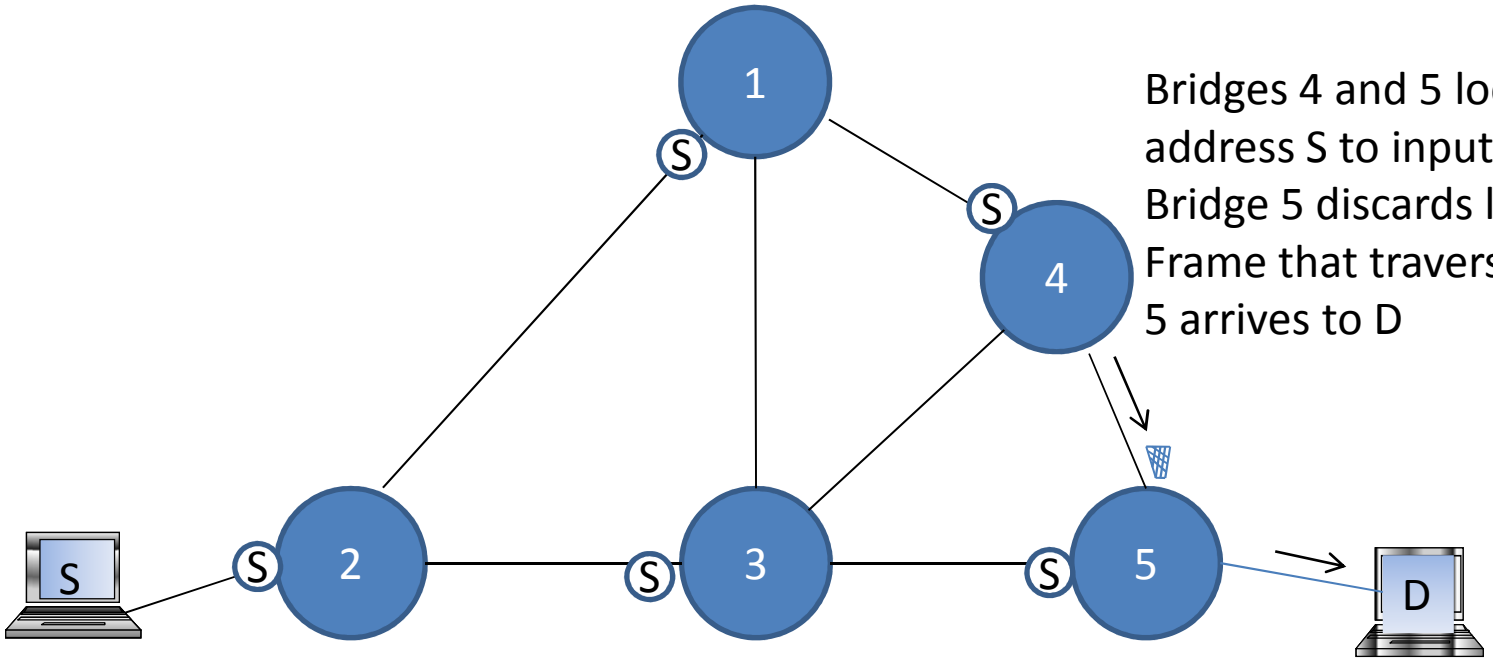
D *Port locked to D*

ARP (path) request (broadcast)

Late frame discarded

Path set up . 4

The fastest ARP Request reaches destination host



Bridges 4 and 5 lock learning of address S to input port
Bridge 5 discards late frame
Frame that traversed bridges 2-3-5 arrives to D

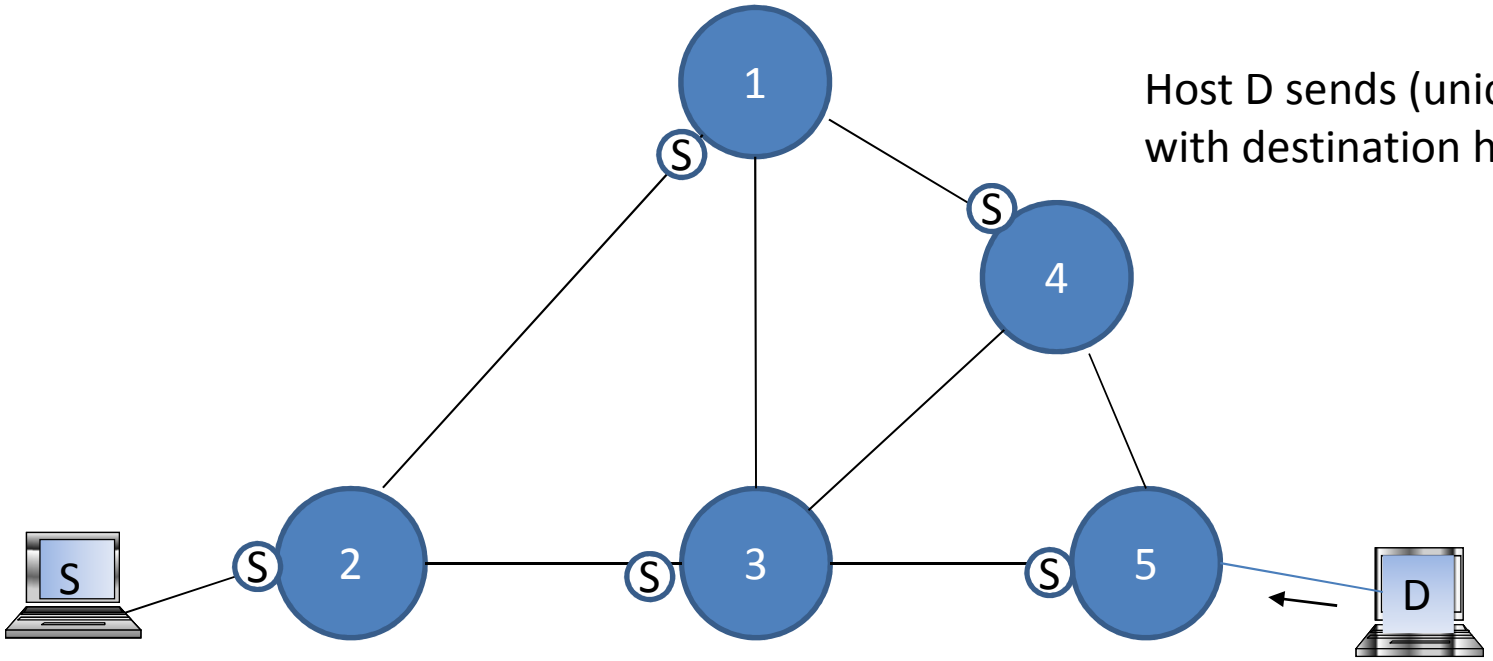
S *Port locked to S*

D *Port locked to D*

→ *ARP (path) request (broadcast)*

← *ARP (path) reply (confirm) (unicast)*

Path set up . 5
ARP Reply (unicast)



Host D sends (unicast) ARP Reply with destination host S

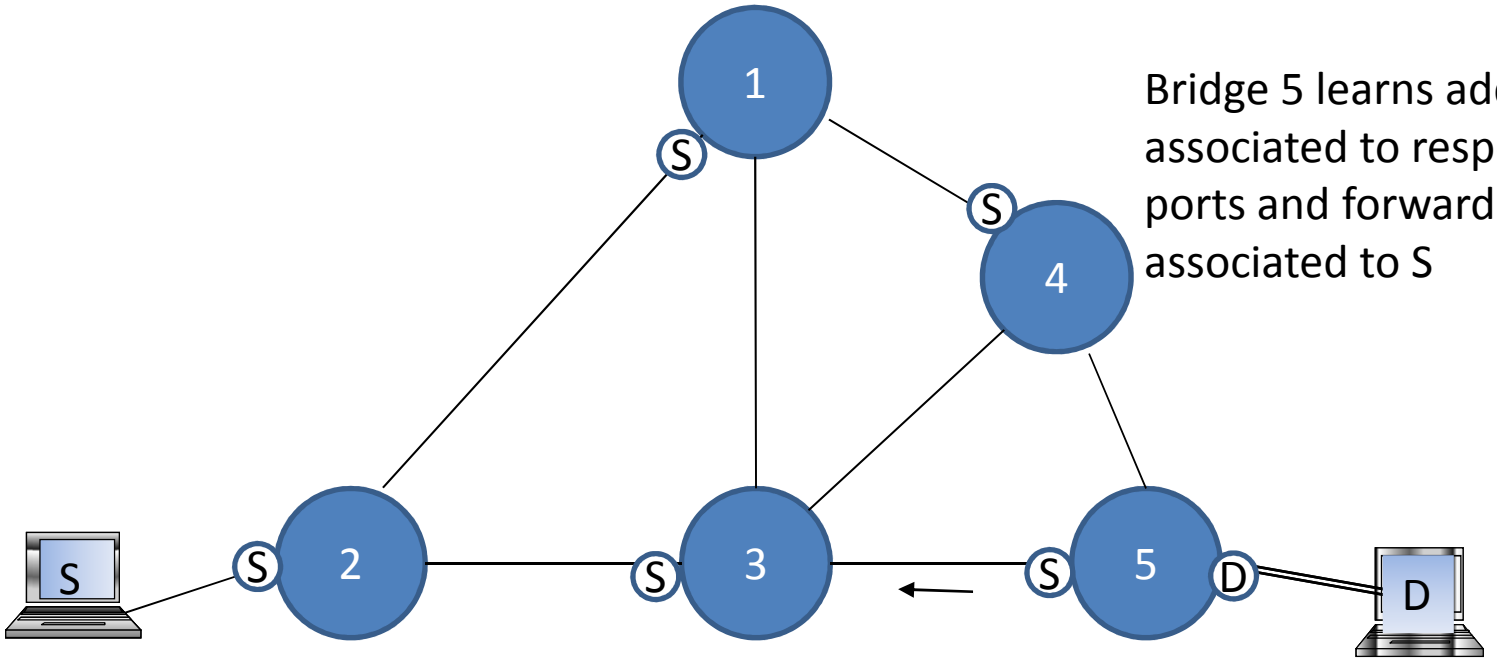
S *Port locked to S*

D *Port locked to D*

→ *ARP (path) request (broadcasted)*

← *ARP (path) reply (confirm) (unicast)*

Path set up . 6
ARP Reply



Bridge 5 learns address D associated to respective input ports and forwards via port associated to S

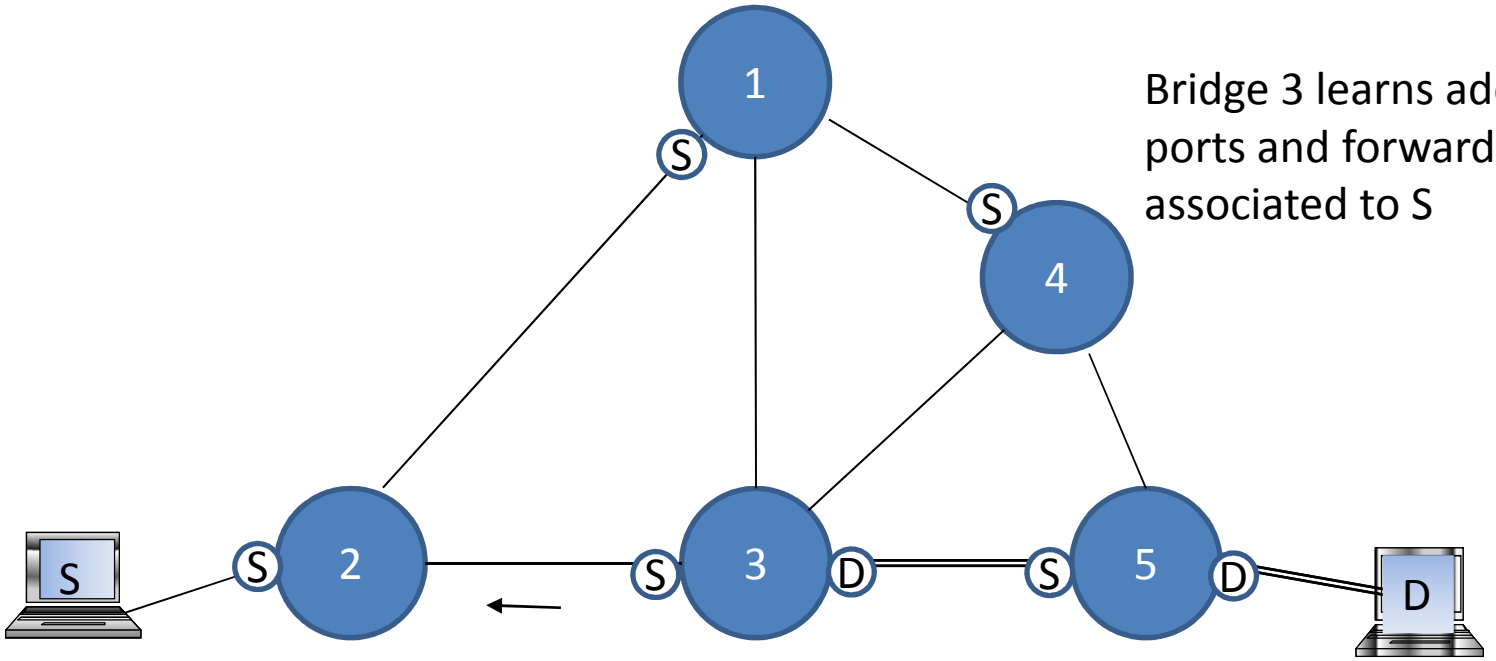
S *Port locked to S*

D *Port locked to D*

→ *ARP (path) request (broadcasted)*

← *ARP (path) reply (confirm) (unicast)*

Path set up . 7
ARP Reply



Bridge 3 learns address D at input ports and forwards via port associated to S

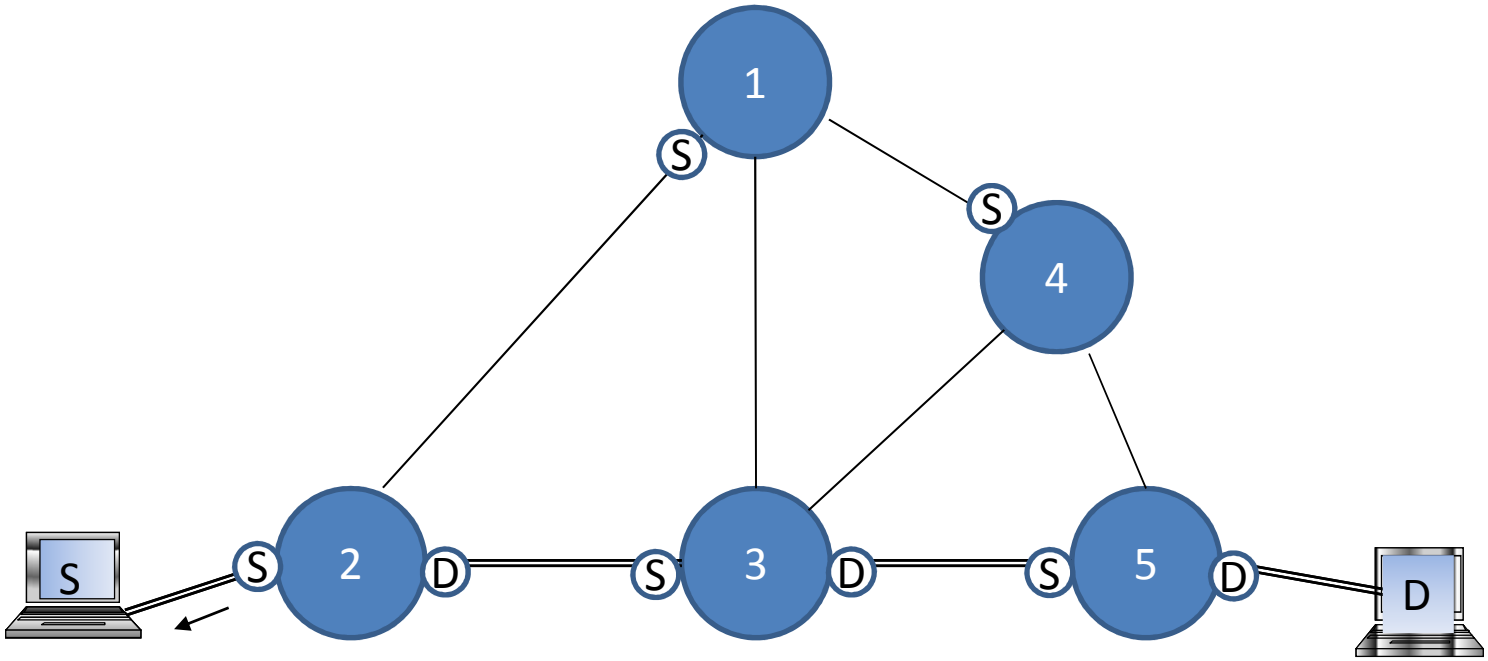
(S) *Port locked to S*

(D) *Port locked to D*

→ *ARP (path) request (broadcasted)*

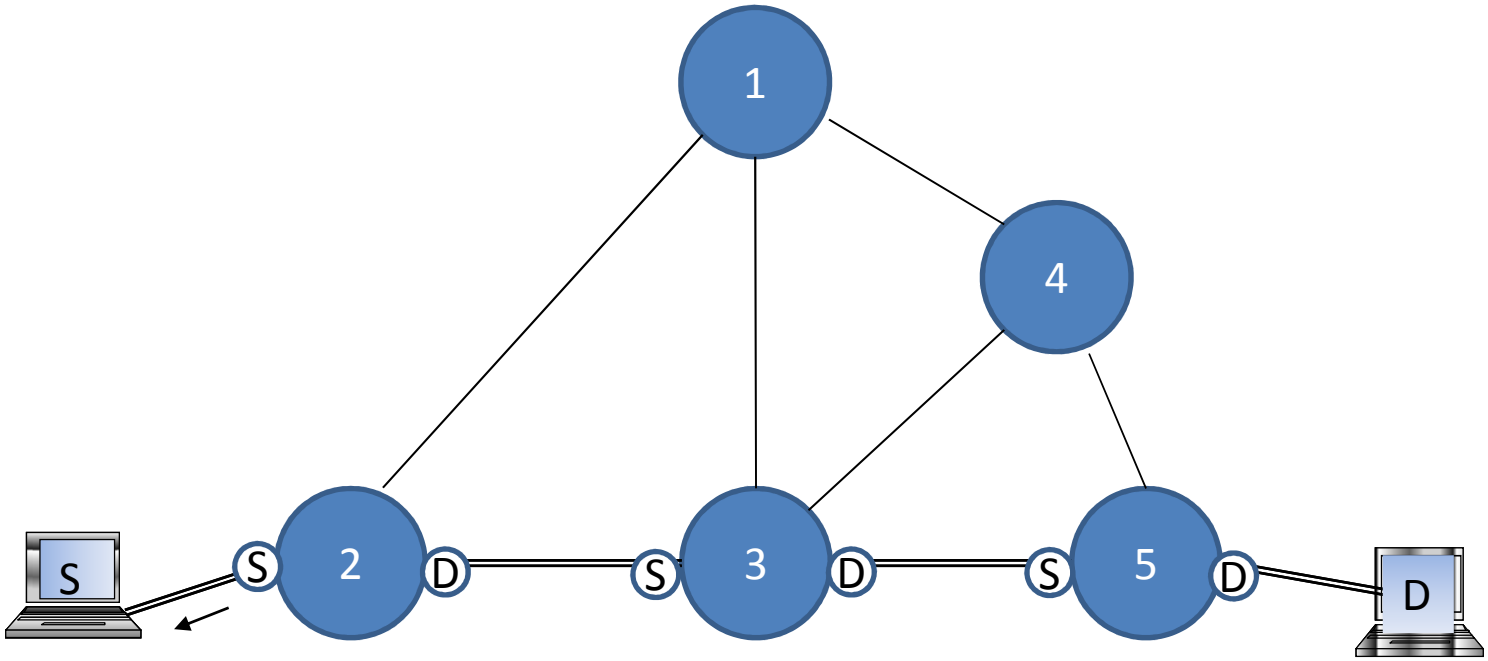
← *ARP (path) reply (confirm) (unicast)*

ARP Reply arrives at S and completes the path set up



A symmetrical path is built between S and D
A temporary tree towards S is built

Other tree branches created, but no confirmed, expire



S *Port locked to S*

D *Port locked to D*

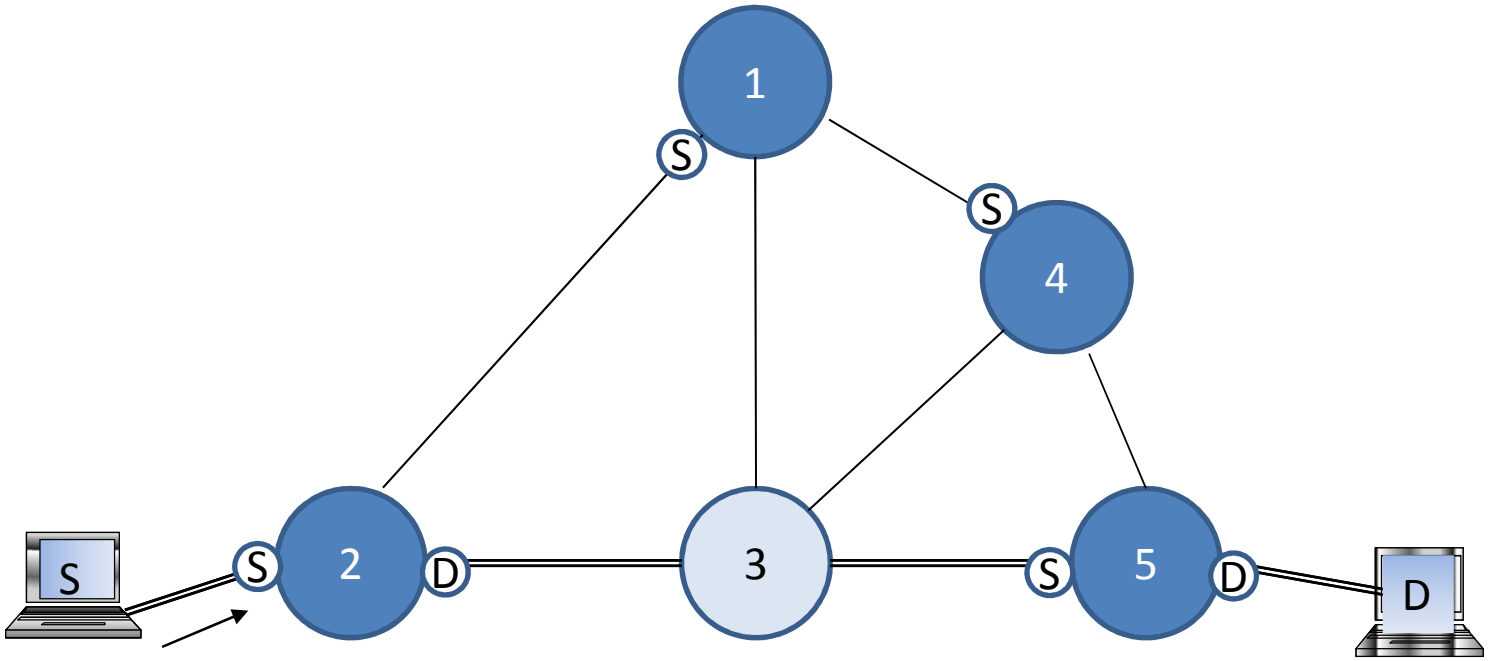
→ *ARP (path) request (broadcasted)*

← *ARP (path) reply (confirm) (unicast)*

Path repair

- A bridge reinitializes, learnt MACs are flushed.
- A unicast frame arrives at a bridge where its destination address is unknown (not associated to any port as source).
- Several variants to repair the path
  - ARP Request reissued from the bridge w/o path
    - Does not work if there are no redundant links in forward direction
  - Encapsulate frame on broadcast frame (with all ARP-Path bridges multicast destination address and return it via input port towards source bridge, who reissues ARP Request.
  - Other variants possible

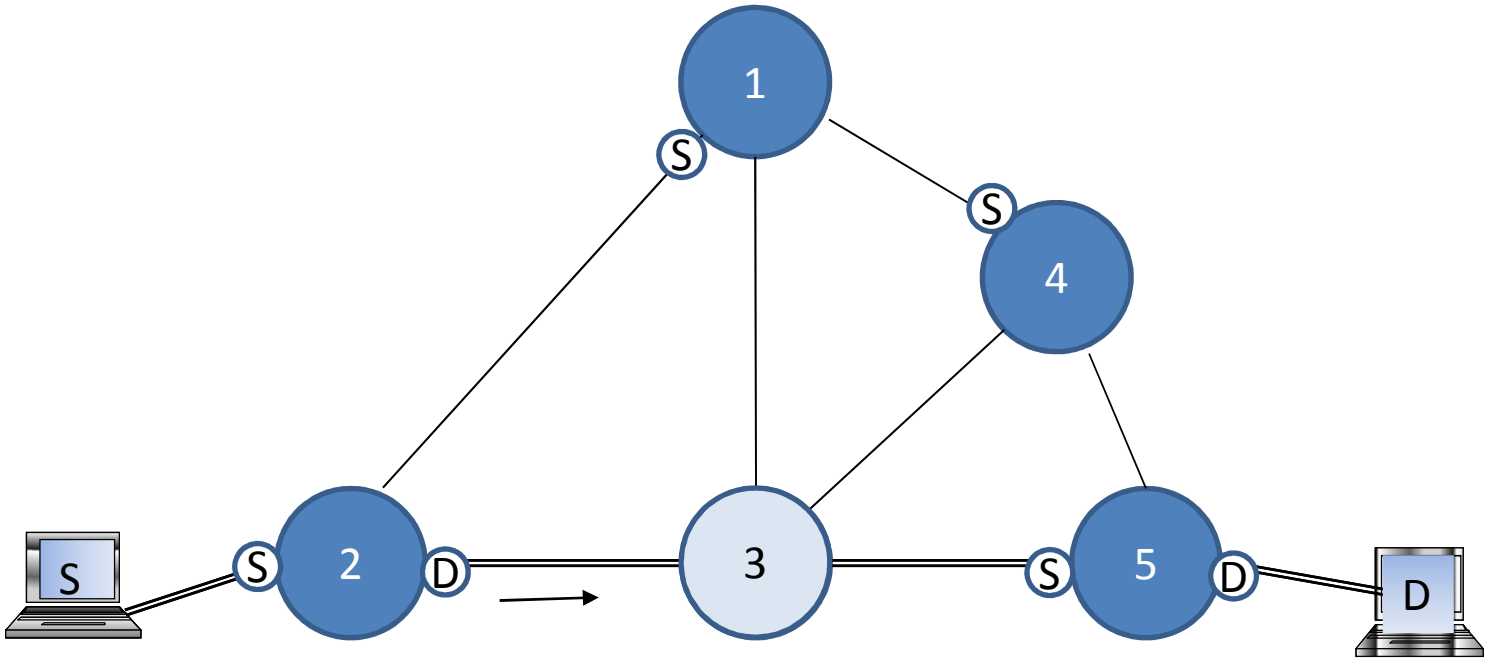
Path repair (bridge 3 flushed all its MACs by initialization after failure)



S *Port locked to S*

D *Port locked to D*

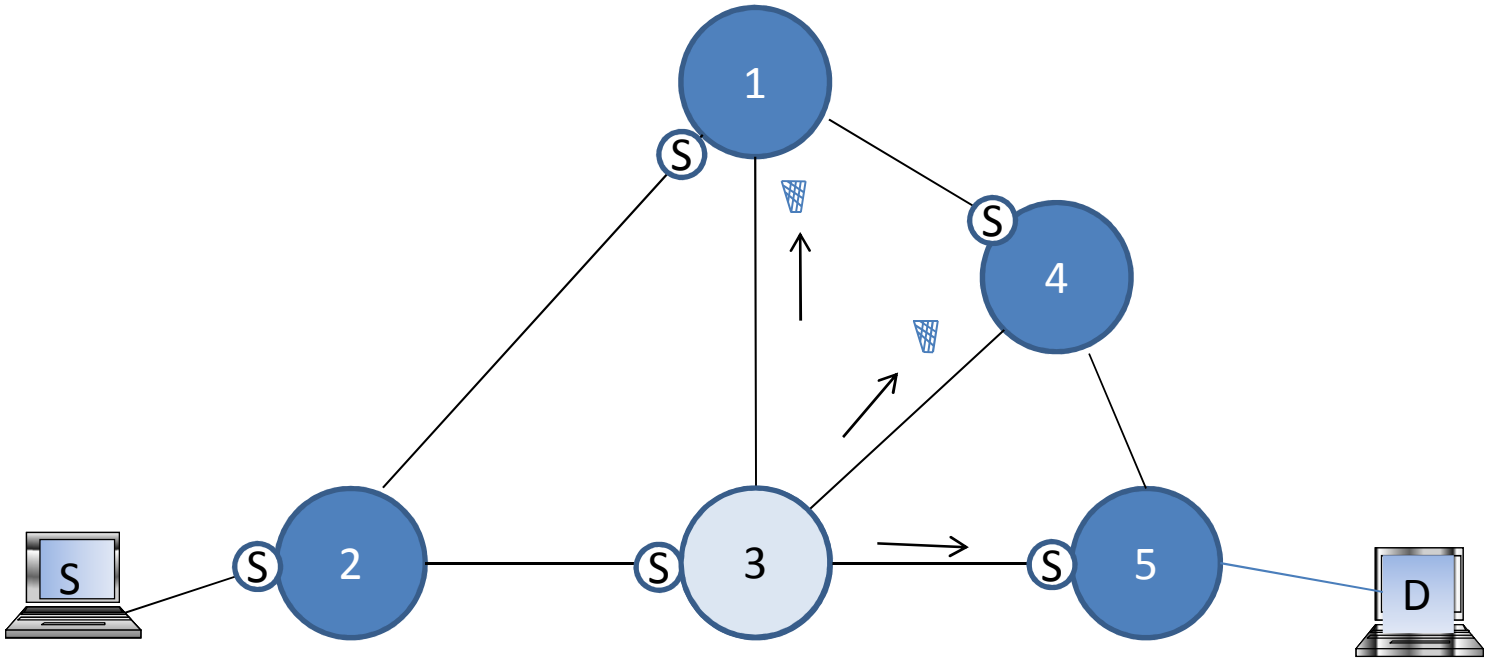
Path repair (bridge 3 had all MACs flushed by initialization)



S *Port locked to S*

D *Port locked to D*

Path repair (bridge 3 issues ARP request)



S *Port locked to S*

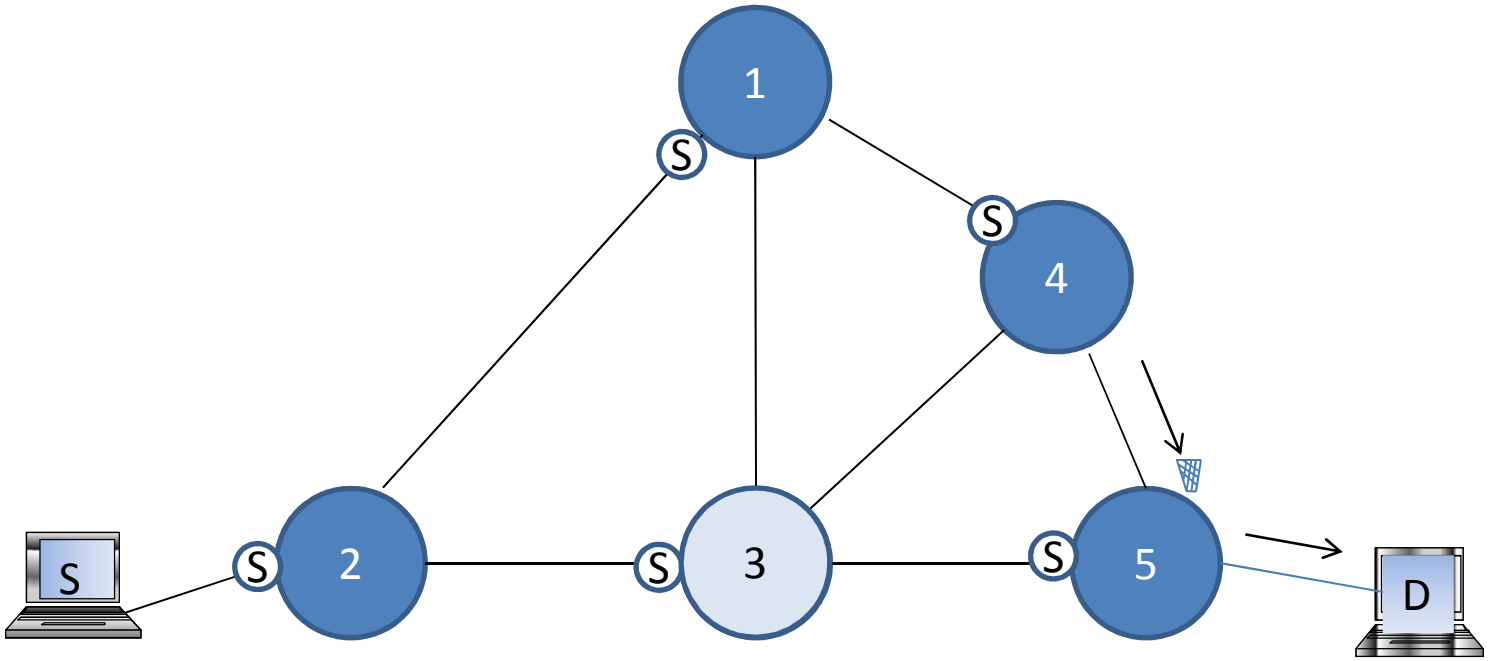
D *Port locked to D*

ARP (path) request (broadcasted)

Late frame discarded

ARP (path) reply (confirm) (unicast)

Path repair



S *Port locked to S*

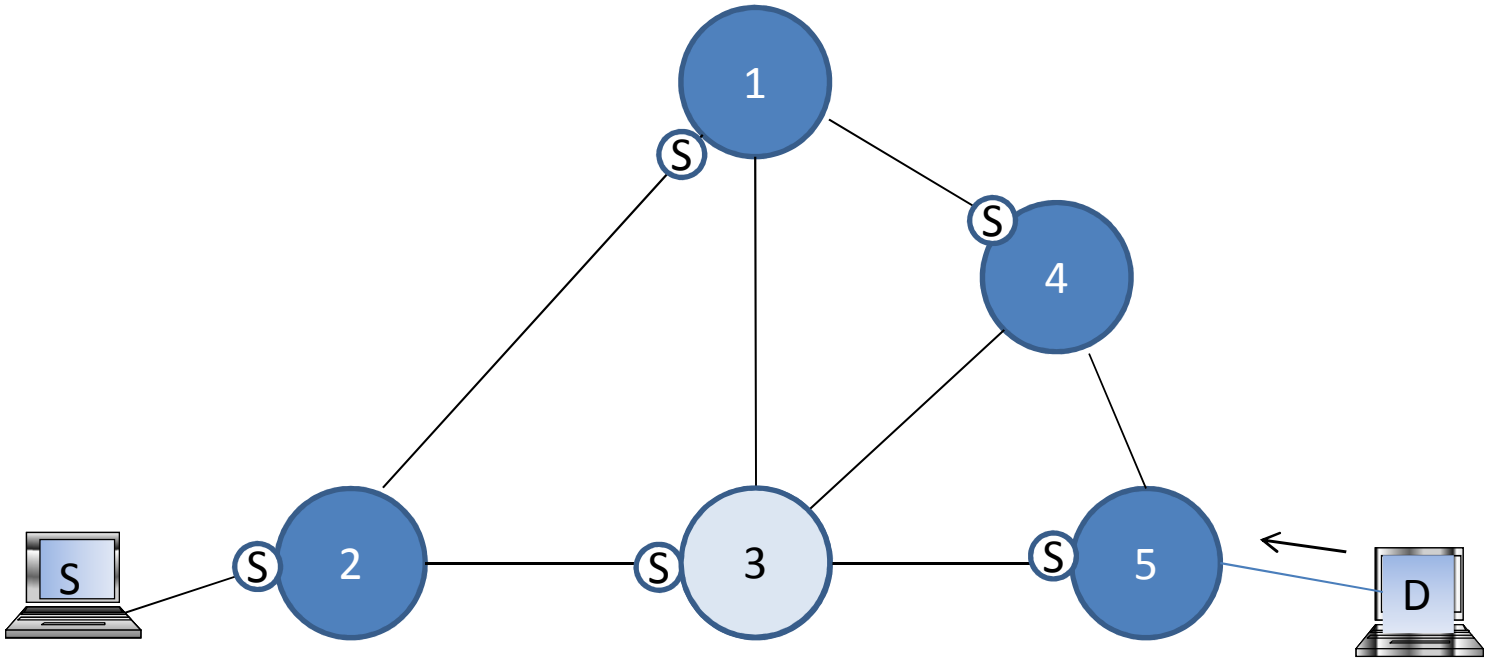
D *Port locked to D*

ARP (path) request (broadcasted)

Late frame discarded

ARP (path) reply (confirm) (unicast)

Path repair completing



S *Port locked to S*

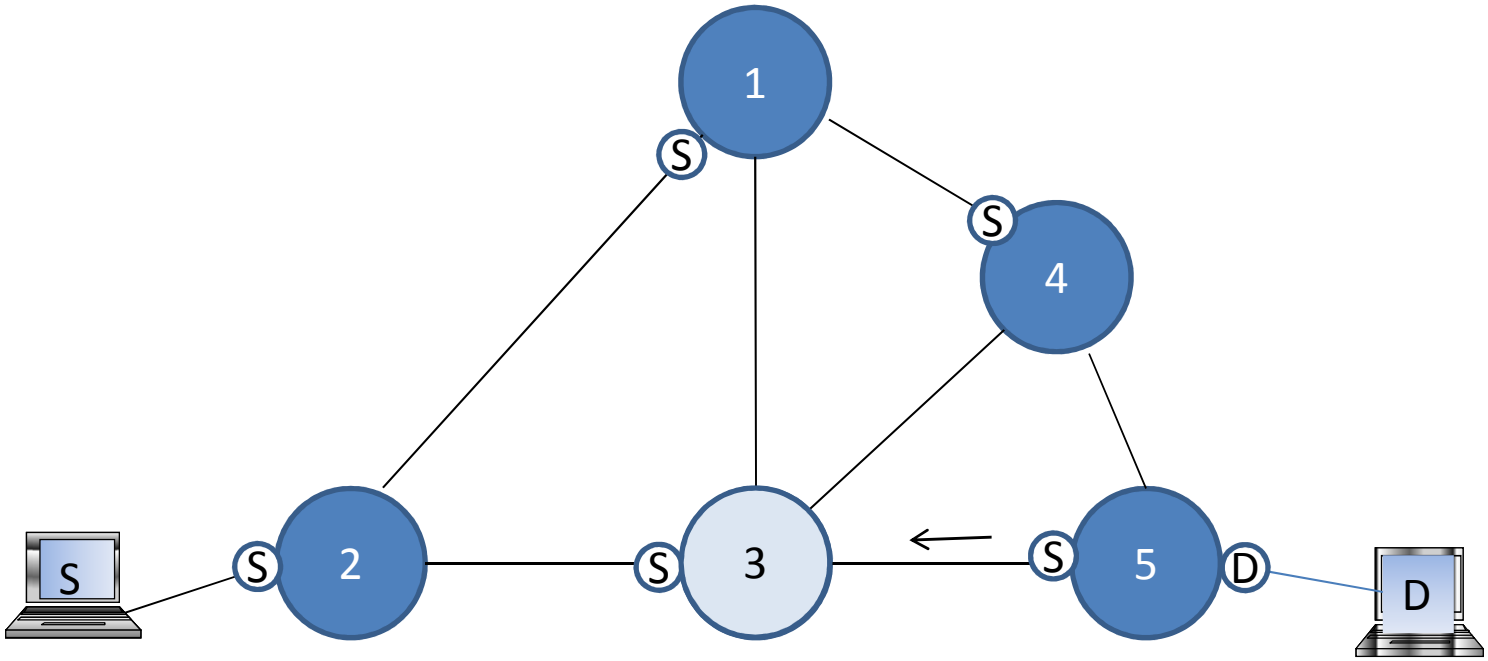
D *Port locked to D*

ARP (path) request (broadcasted)

Late frame discarded

ARP (path) reply (confirm) (unicast)

Path repair completing



S *Port locked to S*

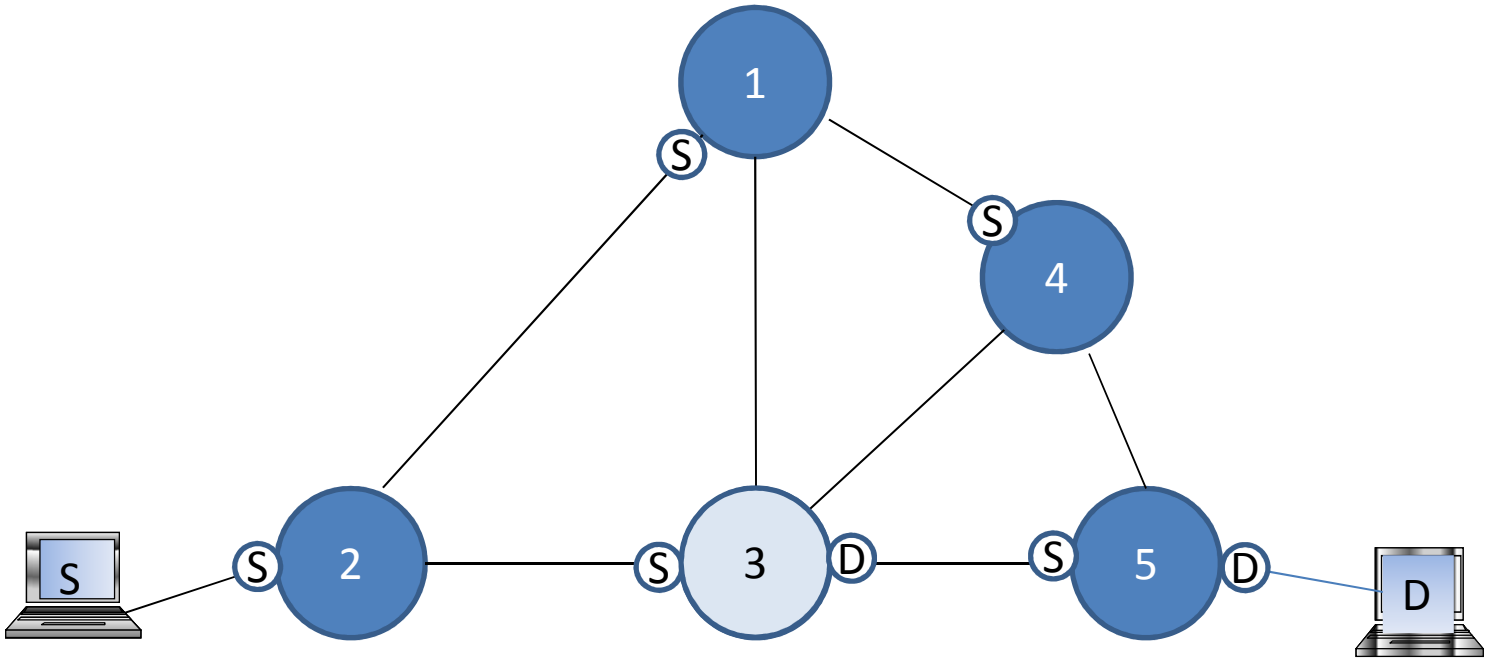
D *Port locked to D*

ARP (path) request (broadcasted)

Late frame discarded

ARP (path) reply (confirm) (unicast)

Path repair completed



S *Port locked to S*

D *Port locked to D*