

Fast path bridges

Old and new ideas for the evolution of transparent bridges

IEEE 802.1 Interim Meeting Sept 2009

Guillermo Ibáñez

GIST research group. Telematic Eng. Area. Dpt. Automatica.

Universidad de Alcalá (Madrid)

guillermo.ibanez@uah.es

Transparent switch evolution

- Standardized:
 - STP, RSTP, MSTP
- Under standardization:
 - Shortest Path Bridges
 - Routing Bridges (TRILL, IETF)
 - *Both use proven link state routing IS-IS to build source rooted trees and path computation*

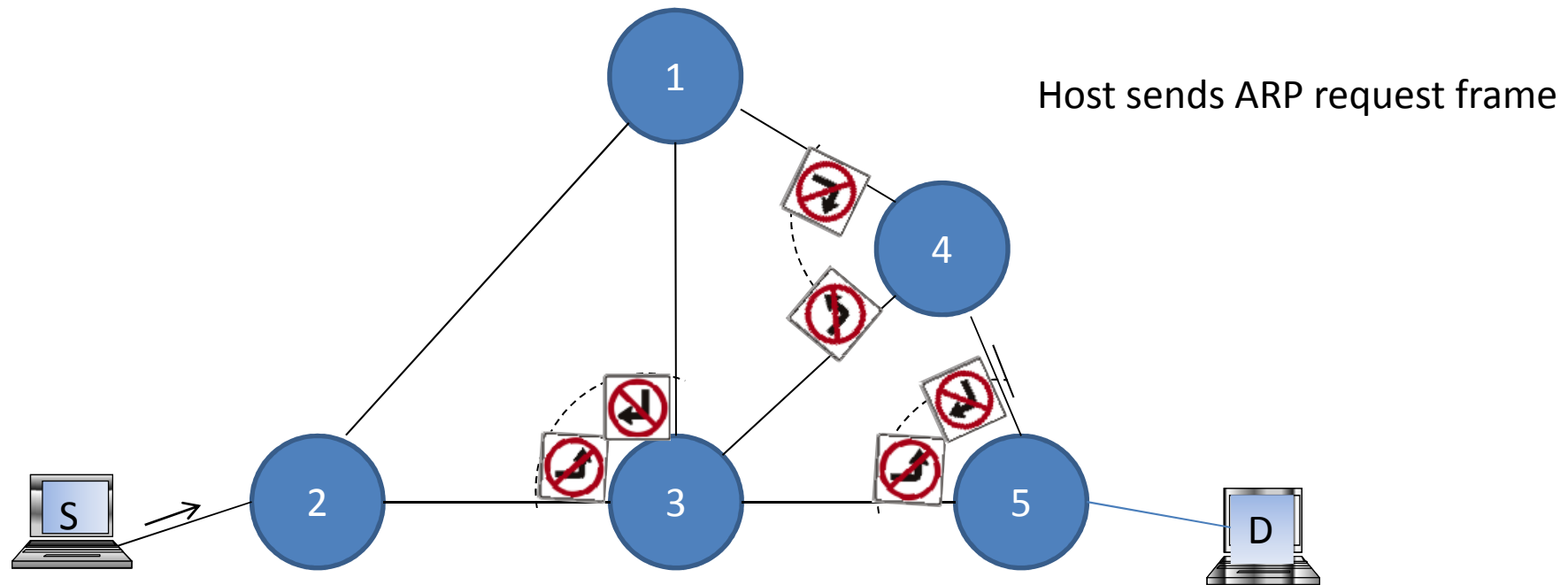
Motivation / Basis

- Shortest path bridges and Routing bridges represent the evolution of the transparent bridge paradigm beyond the limitation of STP and the complexities of configuring efficiently MSTP
- But conceptually SPBs and RBs are still *hybrid devices (bridge + router)*:
 - Combination of transparent bridge and link state routing protocol (IS-IS) is used to build trees and find routes.
- *We think that the evolution of transparent bridges could also be possible by enhancing the transparent bridge mechanisms of forwarding, learning and filtering.*
 - An approach is to relax the restrictions of diffusion of frames to a tree topology:
 - Enable diffusion and learning over all (or most) links : *Fast path*
 - But an effective and simple loop prevention mechanism is mandatory
 - We find Up/Down simple and performant, but does not always provide shortest paths ($\approx 8\%$ to 25% of prohibited turns, best with network topologies with wide degree distribution range)
 - Tree Agreement Protocol requires tree building by cost comparison (distance vector)
 - Other simple and distributed alternative mechanisms for loop prevention?

Fast path basics

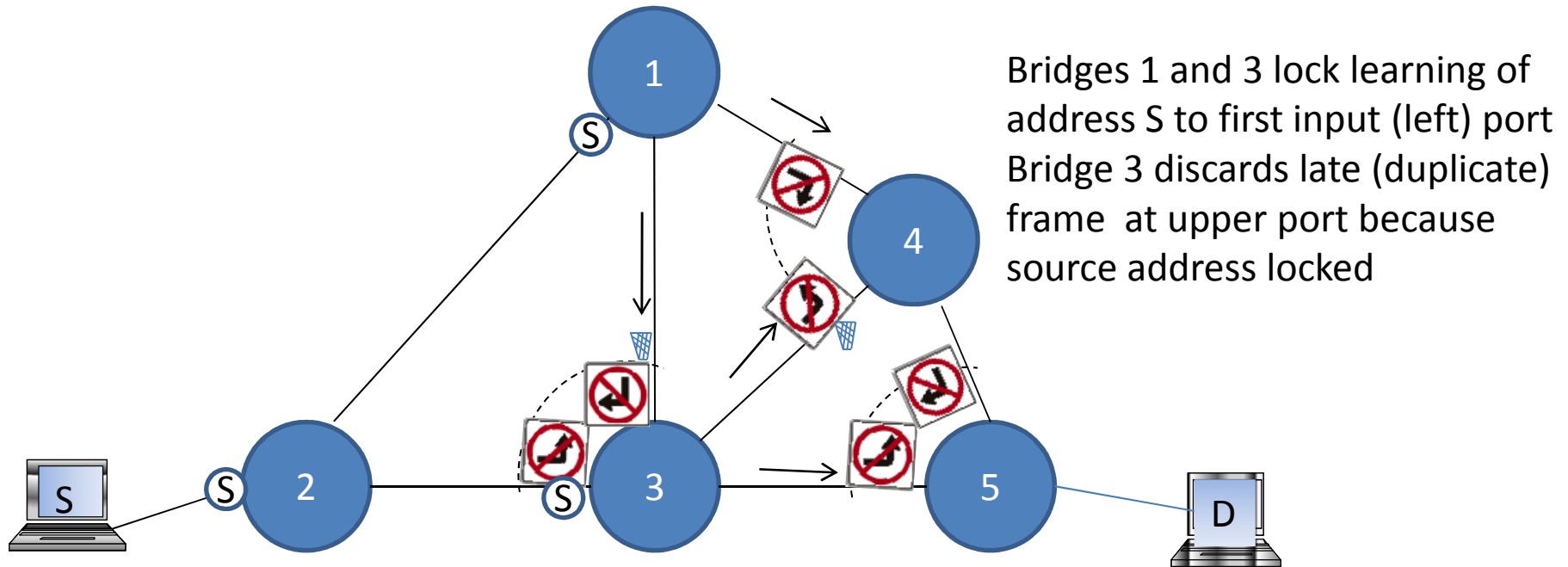
- Establish unicast paths and source bridge rooted multicast trees without ancillary routing protocol, just by controlled flooding (from host or bridges respectively), of a broadcasted frame.
- The fastest path, among the paths permitted by the loop prevention mechanism (Up/Down default), gets selected at every bridge.
- The path is then confirmed as a bidirectional, symmetric path, after reception of the unicast reply frame from destination host or bridge(s).

Operation (path set up by host-1)



- Ⓢ Port locked to S
 - ⓓ Port locked to D
 - ARP (path) request (broadcasted)
 - ← ARP (path) reply (confirm) (unicast)
- Prohibited turn (down-up)

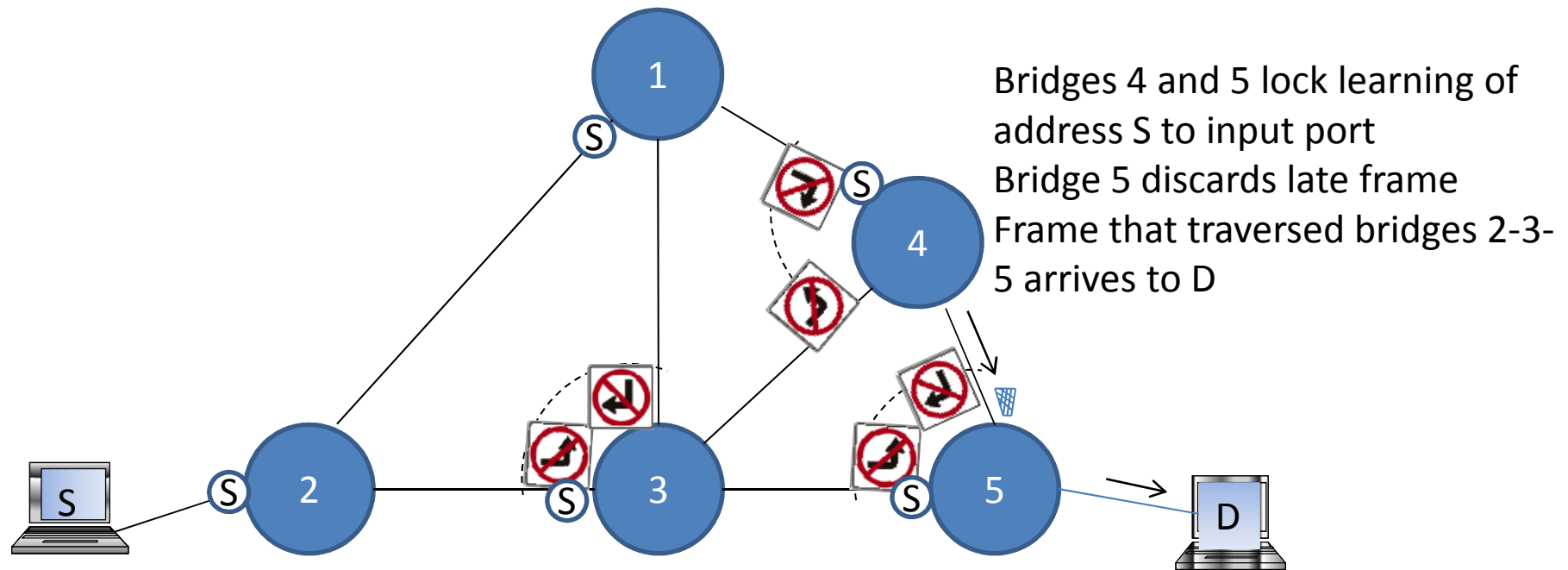
Operation (path set up by host-III)



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

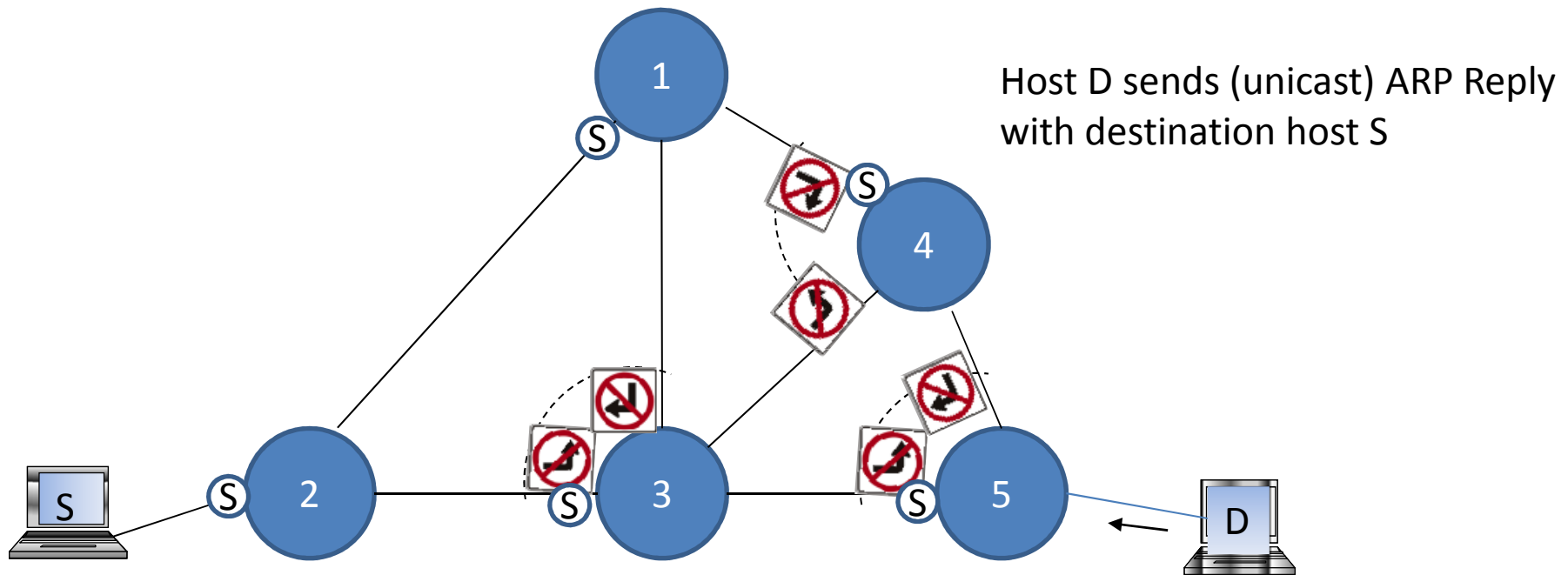
- Ⓢ Port locked to S
- ⓓ Port locked to D
- ARP (path) request (broadcasted)
- ← ARP (path) reply (confirm) (unicast)
- ⊘ Prohibited turn (down-up)
- 🗑 Late frame discarded







Operation (path set up by host-IV)



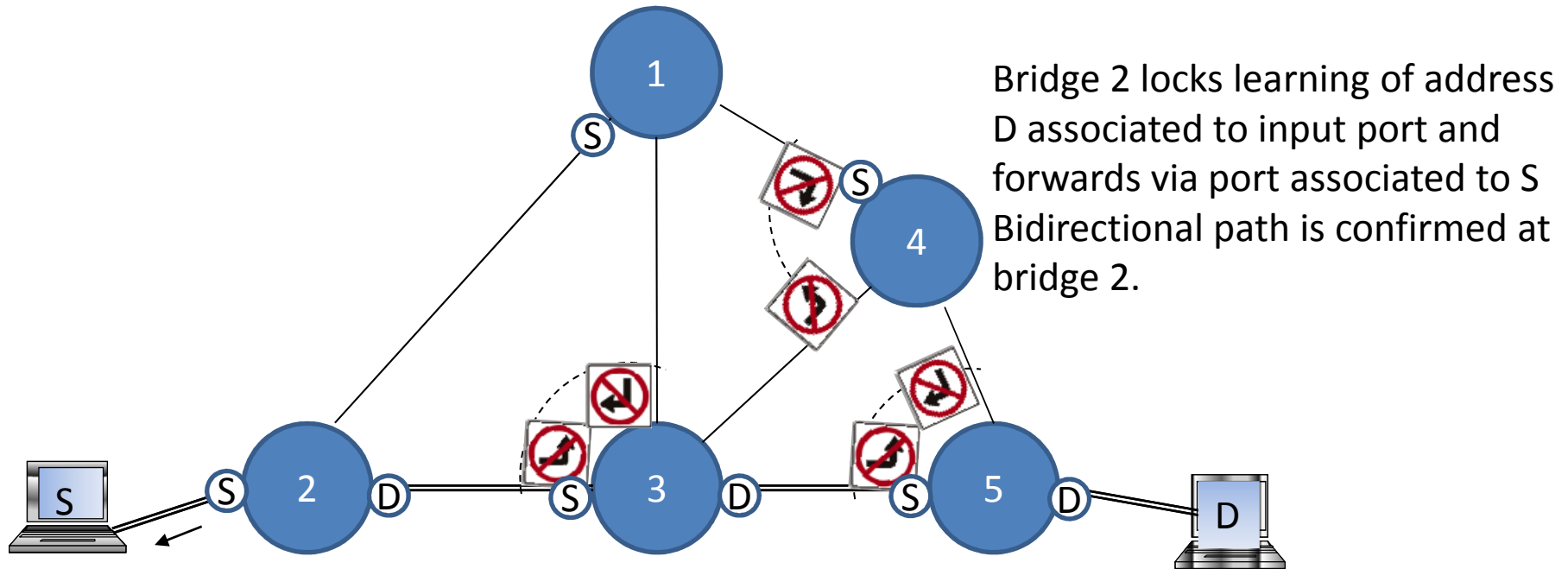
- (S) Port locked to S
 - (D) Port locked to D
 - ARP (path) request (broadcasted)
 - ← ARP (path) reply (confirm) (unicast)
- Prohibited turn (down-up)

Operation (path set up by host-V)





-  Port locked to S
-  Port locked to D
-  ARP (path) request (broadcasted)
-  ARP (path) reply (confirm) (unicast)
-  Prohibited turn (down-up)
-  Prohibited turn (up-down)

Operation (path set up by host VIII)



Bridge 2 locks learning of address D associated to input port and forwards via port associated to S
 Bidirectional path is confirmed at bridge 2.

- Ⓢ Port locked to S
 - ⓓ Port locked to D
 - ARP (path) request (broadcasted)
 - ← ARP (path) reply (confirm) (unicast)
-  Prohibited turn (down-up)



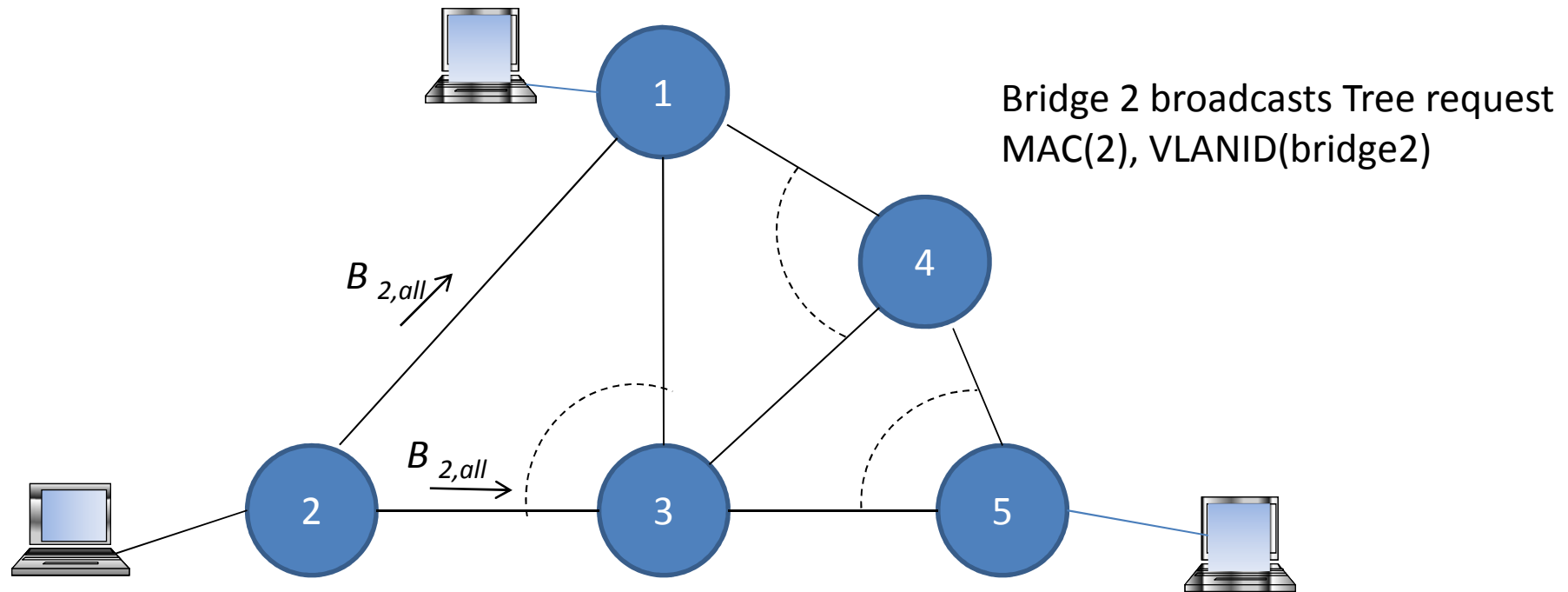
Pseudocode

Pseudocode

Frame received at bridge

- *destination address is broadcast or multicast*
 - *source address is unknown*
 - *Associate source address to input port, block learning of this address at other ports*
 - *Activate cache timer, wait for path confirmation in opposite direction by destination*
 - *source address is known (Fastpath exists)*
 - *discard frame if input port is not the Fastpath associated port*
 - *forward frame through all ports except prohibited turns and refresh persistency timer of source address*
- *destination address is unicast*
 - *Destination address is known*
 - *Frame received at associated port (i.e. source address is associated to the port which received the frame)*
 - » *Forward to output port associated to destination address*
 - » *Confirm /refresh bidirectional Fastpath source-destination association*
 - *Frame received at another port (i.e. source address is associated to a different port)*
 - » *Discard frame*
 - » *Count event*
 - *Destination address unknown*
 - *Send broadcast Path Request via all permitted output ports*
 - *Wait for Path Reply from bridge parent of destination bridge*
 - » *Path Reply received : Send unicast frame*
 - » *No Path Reply: Send Path Reject backwards towards source address if no reply.*
 - » *Designated bridge of source intercepts the Path Reject and broadcasts a Path Request to set up a new path to destination*

Source rooted Tree set up by bridge – I



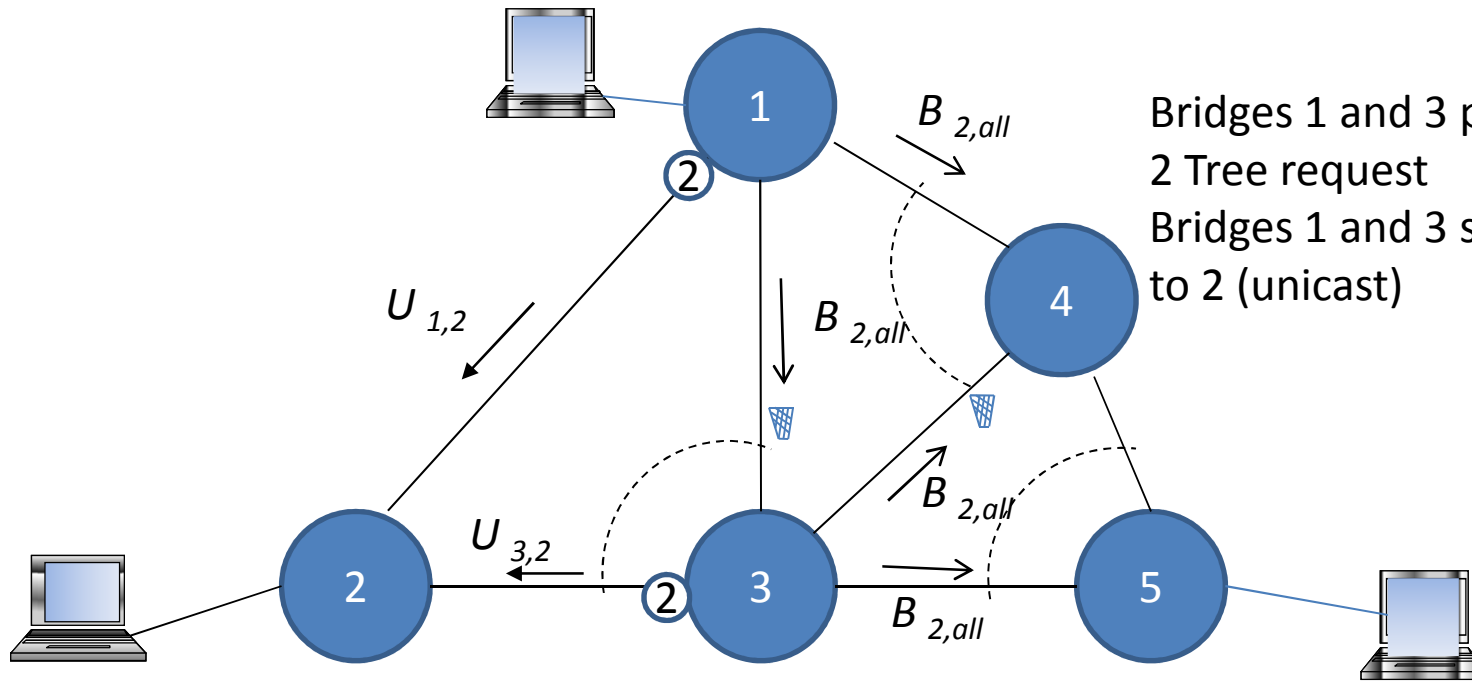
② Port associated to tree
rooted at bridge 2

Prohibited turn (down-up)

→ Tree path request
(broadcasted)

← Tree path reply (confirm) (unicast)

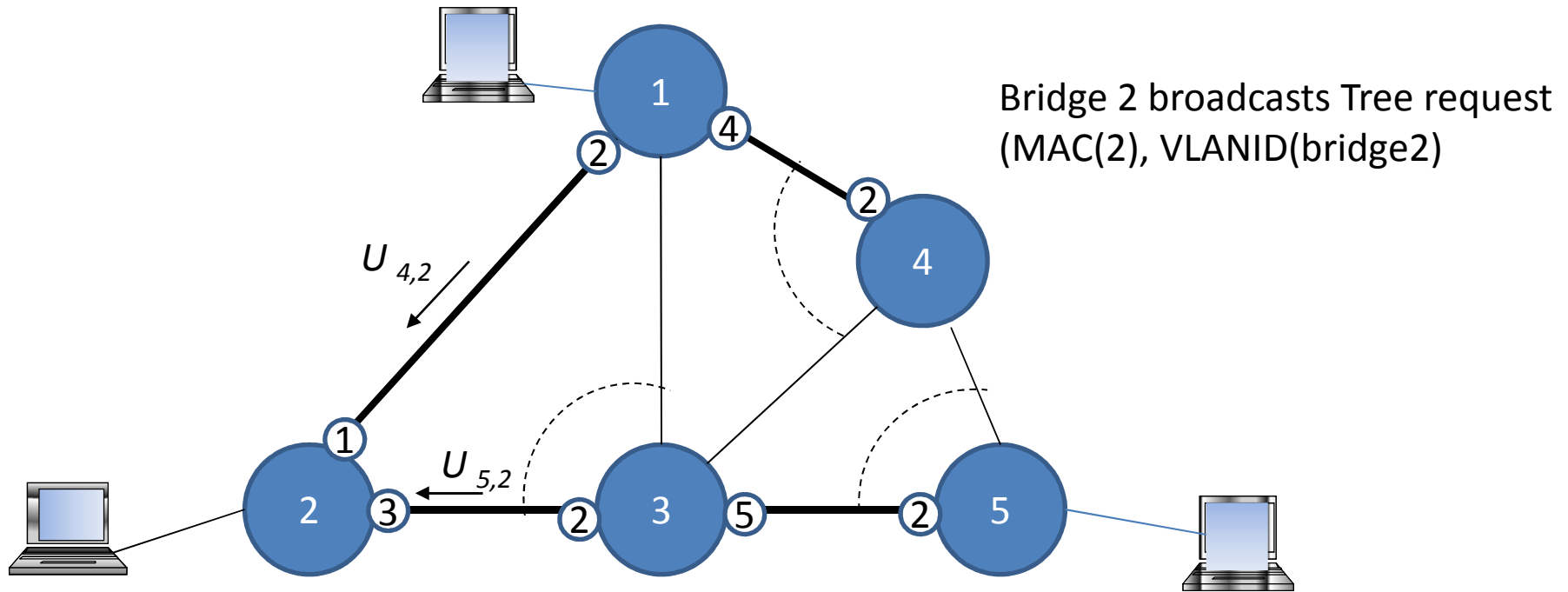
Tree set up by bridge -II)



Bridges 1 and 3 propagate bridge 2 Tree request
 Bridges 1 and 3 send tree confirm to 2 (unicast)

- ② Port associated for tree rooted at bridge 2
- Tree path request (broadcasted)
- ← Tree path reply (confirm) towards root bridge (unicast)
- Prohibited turn (down-up)
- Late frame discarded

Tree set up by bridge -IV)



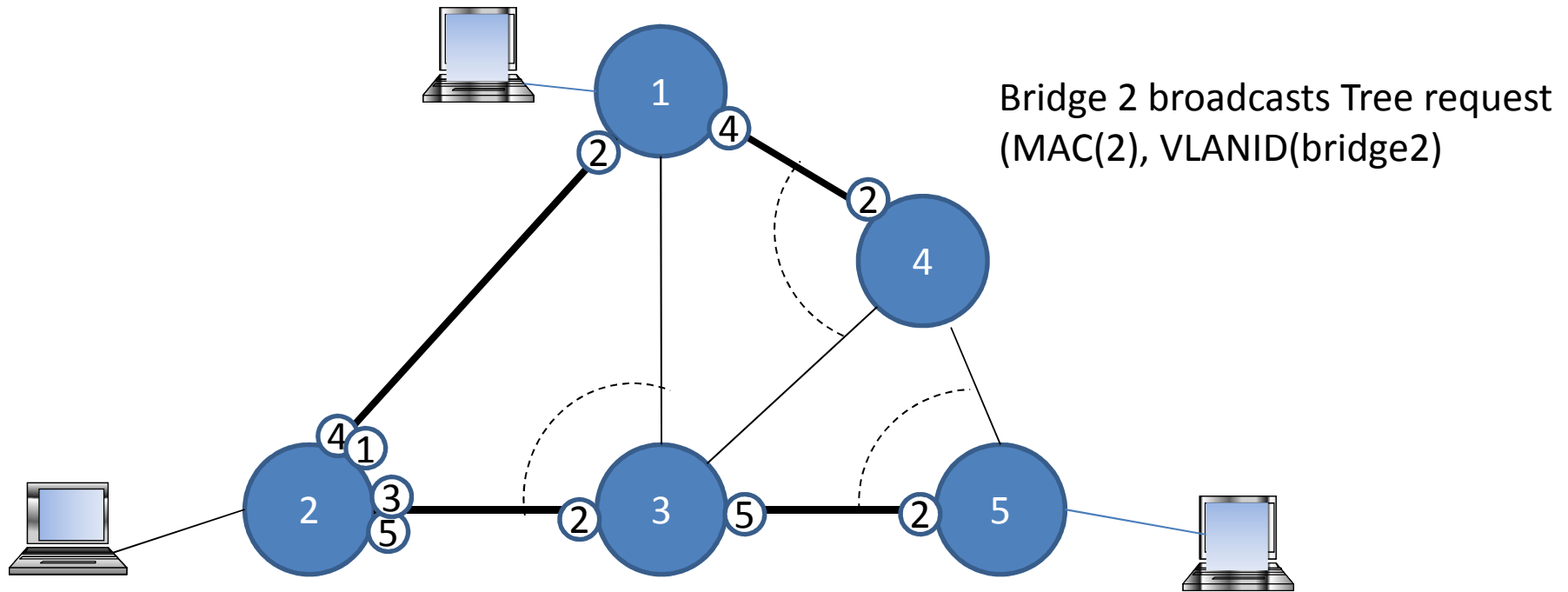
② Port associated to tree rooted at bridge 2

Prohibited turn (down-up)

→ Tree path request (broadcasted)

← Tree path reply (confirm) (unicast)

Tree set up by bridge -V)



② Port associated to tree rooted at bridge 2

Prohibited turn (down-up)

→ Tree path request (broadcasted)

← Tree path reply (confirm) (unicast)

Fast PathTrees construction

- Periodic broadcast of beacon frames from root bridges to keep alive tree paths
 - Tree path expires by link failure or timer expired
 - Stability of path is the priority: change only on path failure

Summary differences SPB vs. Fastpath

Function, feature (*)	SPB (*)	FastpathUD or Fastpath[TBD]
Forwarding paths obtained	Shortest paths SP (optimum)	Fastest permitted path FPP (near, not optimum)
Unicast and multipath paths requirement	Congruency required	Idem
Compatibility with other IEEE802.1 protocols	Compatibility RSTP, MSTP, identify Shortest Path Tree (SPT) regions	Idem
Source Trees obtained	Obtains Shortest Path Trees (SPTs)	Obtains Fastest permitted Path Trees (FPPTs)
Identification of Source Tree for SPB application (small medium network size)	SPB: Identify SPT by SPVID	Idem
End station location, SPB application. environment	SPB: end node MAC learning	Idem
Loop prevention	Tree Agreement Protocol	Up/Down or other (TBD)
Loop mitigation	Ingress checking	Idem
Port Forwarding control (synch.)	Tree Agreement Protocol (TAP)	RSTP based

(*) Extracted from from D. Fedyk's presentation

Conclusions

- *It seems conceptually consistent to explore solutions that rely more on the data plane to find paths and build source routing trees than on routing protocols.*
- *Controlled flooding is an approach for evolution of bridging without ancillary link state routing*
- *Loop prevention mechanism required*
 - *Up/Down is simple and performant, near shortest paths, but not full shortest paths*
 - *Other mechanisms?*
- *We are open to suggestions and any form of collaboration.*

Some references

- Our Up/Down performance evaluation on different topologies can be seen at : <http://dx.doi.org/10.1016/j.comnet.2009.08.007> (paper deals with other protocol proposal, but values for Up/Down are applicable equally to FastpathUD.)
- Precedent proposal SDS (no point to point, multipoint links allowed: several bridges must elect the one in charge, high complexity and increased oscillations):
B. Rajagopalan; M. Faiman; Load sharing and shortest-path routing in transparently interconnected local area networks. Proceedings Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 1991. IEEE 7-11 April 1991. Vol. 3, pp:1135 - 1144