Using LLDP as a protocol carrier

Version 1

Norman Finn

Cisco Systems

References

- This presentation is: http://www.ieee802.org/1/files/public/docs2009/ab-nfinn-lldp-as-protocol-carrier-1109-v01.pdf

Summary of LLDP

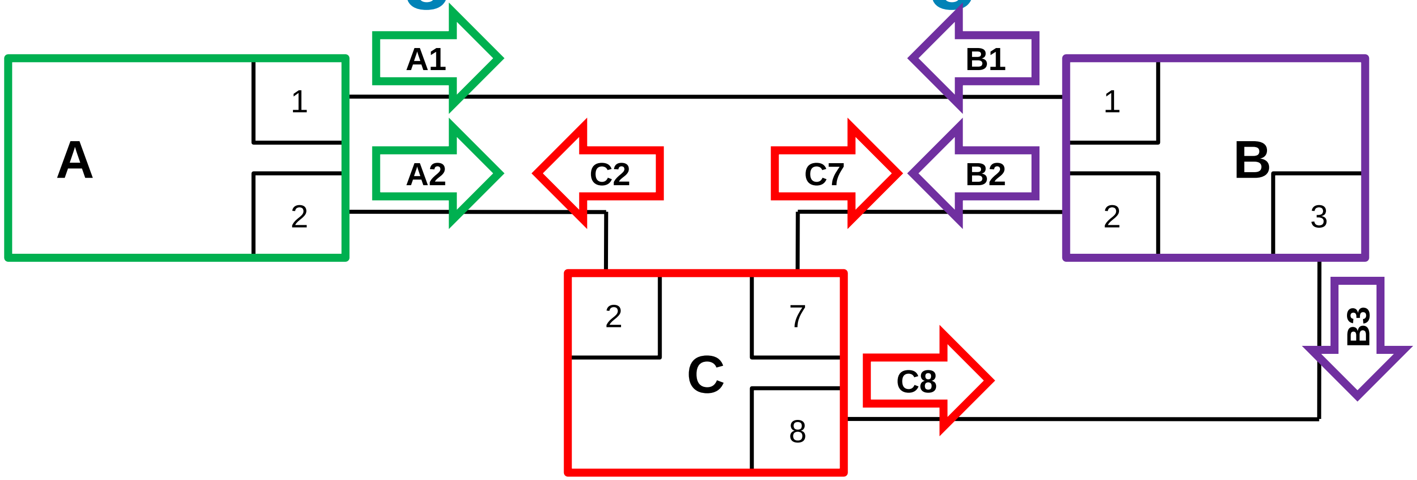
Purpose

- The Link Layer Discovery Protocol, defined in IEEE 802.1AB-2009, is based on the Cisco Discovery Protocol (CDP), invented by Keith McCloghrie.
- Its original, and still primary, purpose is:
  - To advertise the identity of the system and port on that system from which the LLDPDU (LLD Protocol Data Unit) was transmitted; and
  - To collect the information received in LLDPDUs and place it in a MIB, indexed by receiving port, for access by the network manager.
- These two actions enable a network management application (outside the scope of IEEE 802.1) to construct a map of the connectivity of a network.

Extensibility

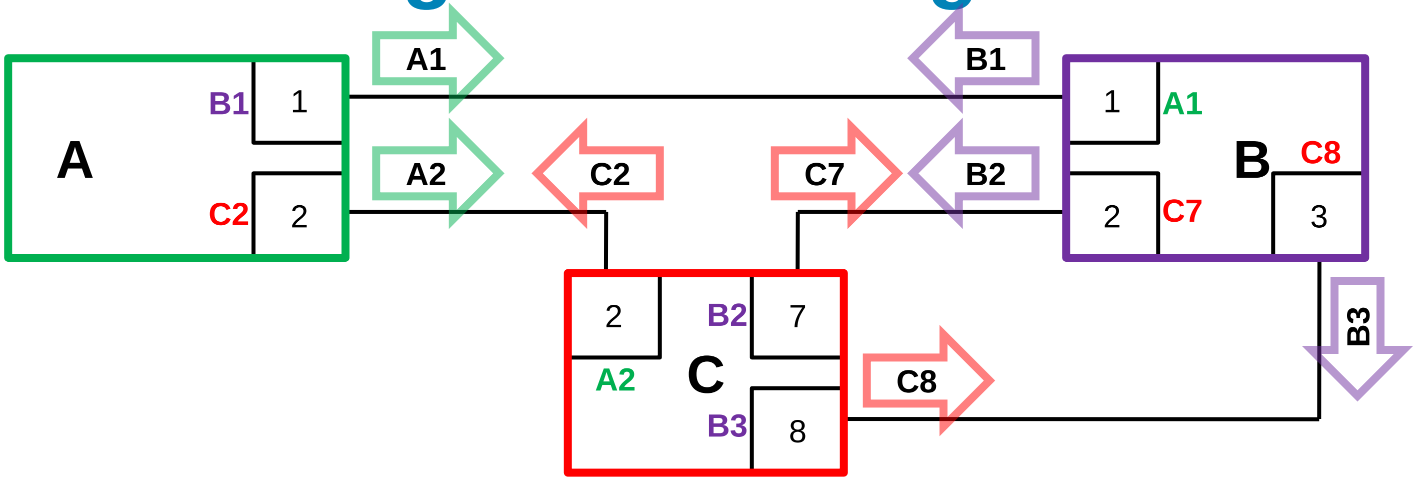
- LLDP is not restricted to bridged networks. Routers, end stations, or other devices are free to use LLDP over any 802-like medium, whether physical or virtual.
- LLDP is extensible, in that its information elements are encoded in independent Type Length Value (TLV) formats within the LLDPDU.
  - — A receiver simply skips over TLVs it doesn’t understand, and processes the TLVs it does understand;
  - — IEEE 802 can extend LLDP by defining new TLVs;
  - — In fact, any entity owning an Organization Unique Identifier (OUI, supplied by the IEEE Registration Authority) can extend LLDP by defining new TLVs.

Discovering nearest neighbors



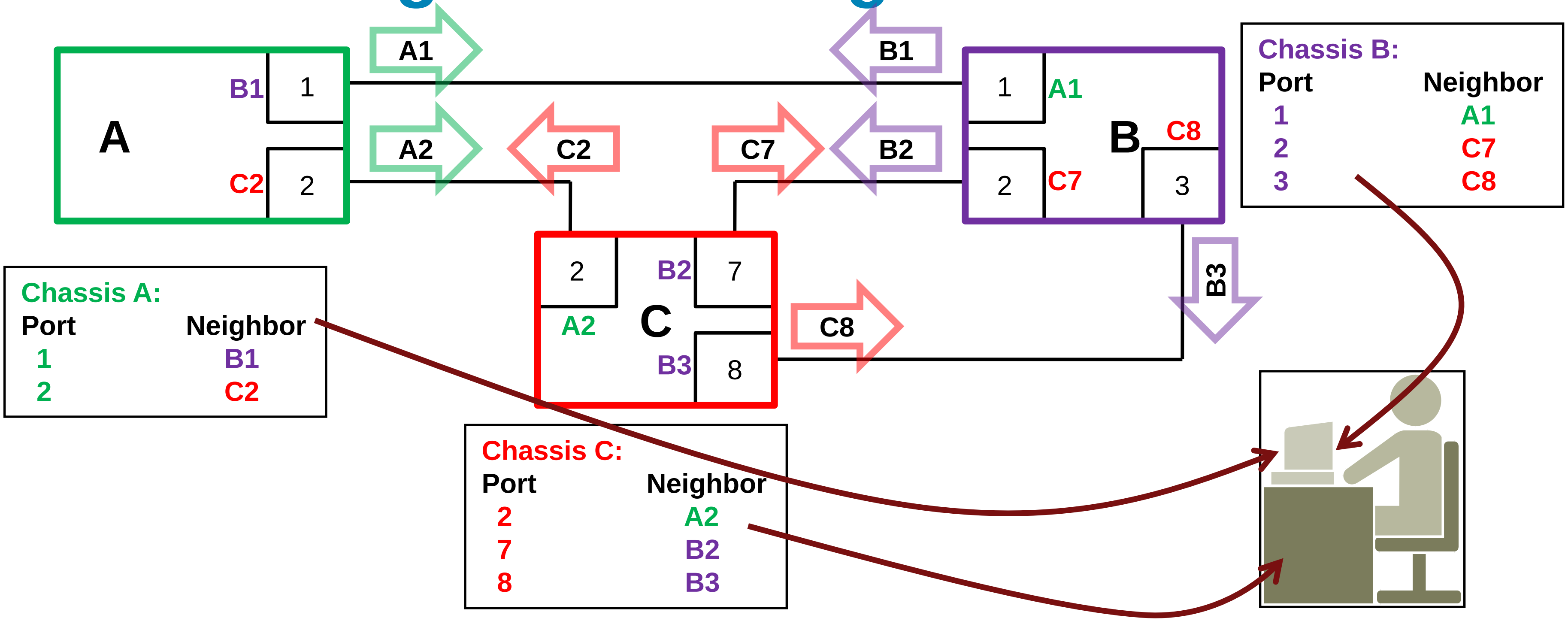
- Each system (**A**, **B**, **C**) advertises itself on each Port (**1**, **2**, ...).

Discovering nearest neighbors



- Each system (**A**, **B**, **C**) advertises itself on each Port (**1**, **2**, ...).
- Each system records its information.

Discovering nearest neighbors



Chassis A:

Port	Neighbor
1	B1
2	C2

Chassis B:

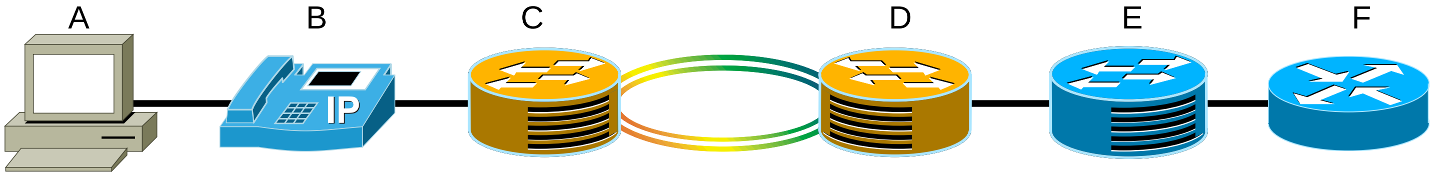
Port	Neighbor
1	A1
2	C7
3	C8

Chassis C:

Port	Neighbor
2	A2
7	B2
8	B3

- Each system (**A**, **B**, **C**) advertises itself on each Port (**1**, **2**, ...).
- Each system records its information.
- Network manager collects information.
  - The individual systems do not have a view of the network – only the network manager can see the whole connectivity picture.

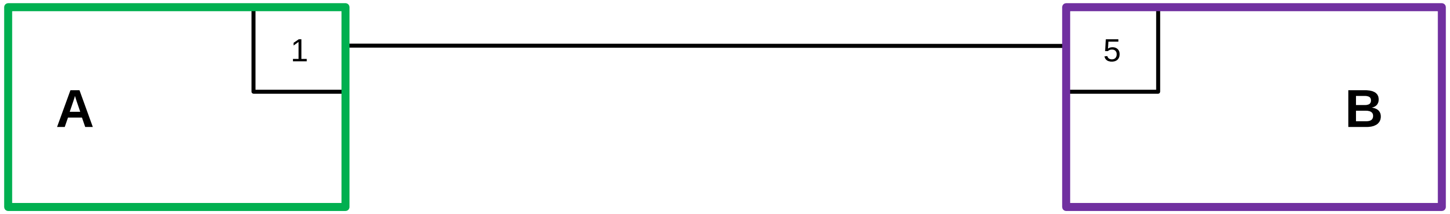
Multiple neighbors at different reaches



Addresses:	A	B	C	D	E	F
Nearest:	Nr B	A Nr C	B Nr D	C Nr E	D Nr F	E Nr
Non-TPMR:	~T C		A ~T D	C ~T E	D ~T F	E ~T
Customer:	Cs E				A Cs F	E Cs

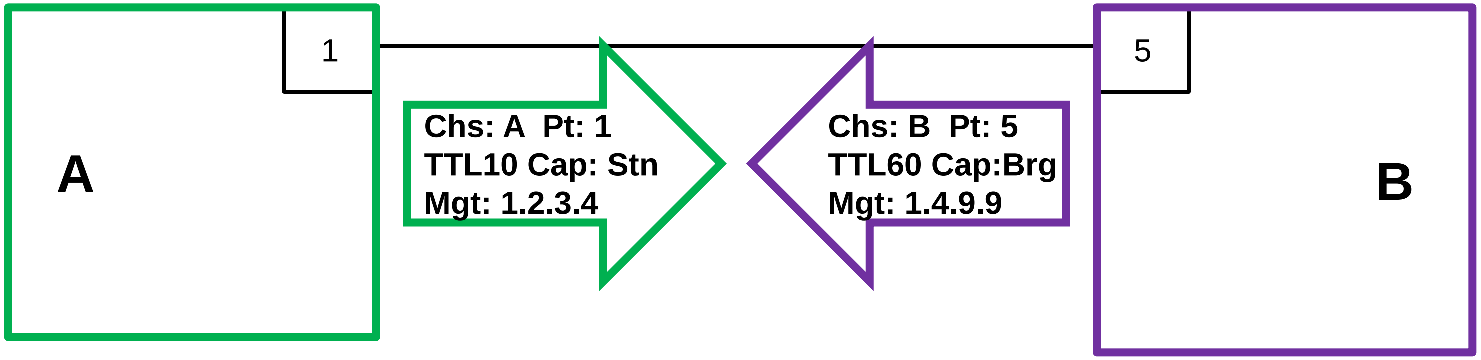
- LLDP can run on multiple destination addresses. A bridge peers, or transparently relays, each address.
- “Reach” of LLDP (via addressing) corresponds to L2 sublayers, so one can have multiple LLDP neighbors:
  - — Physical: Two Port MAC Relay (TPMR) or IP phone (B).
  - — Non-TPMR: Provider bridge carrying an Ethernet service (C,D).
  - — Customer: 802.1Q bridge on other side of provider network (E).
  - — LLDP does not go any further (F).

The LLDP model of operation



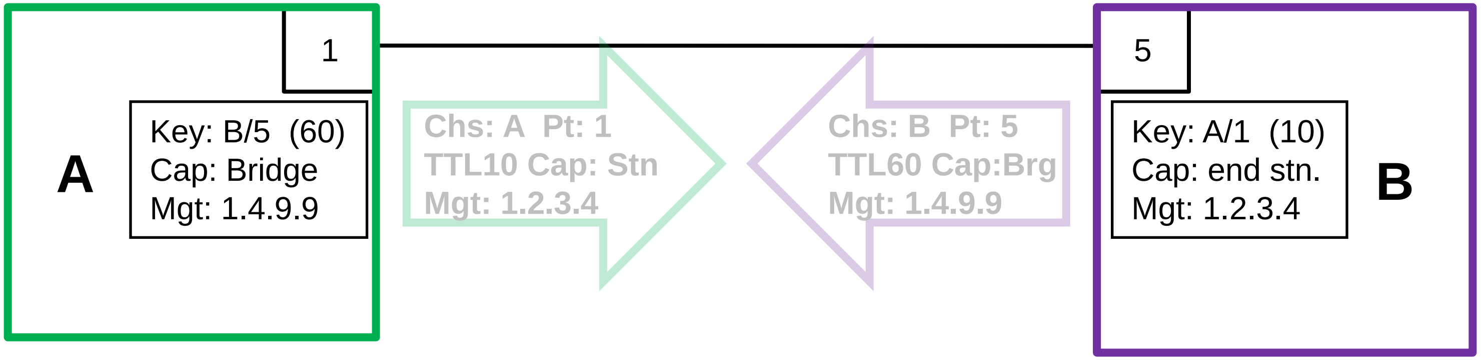
- I’m alone in the world.

The LLDP model of operation



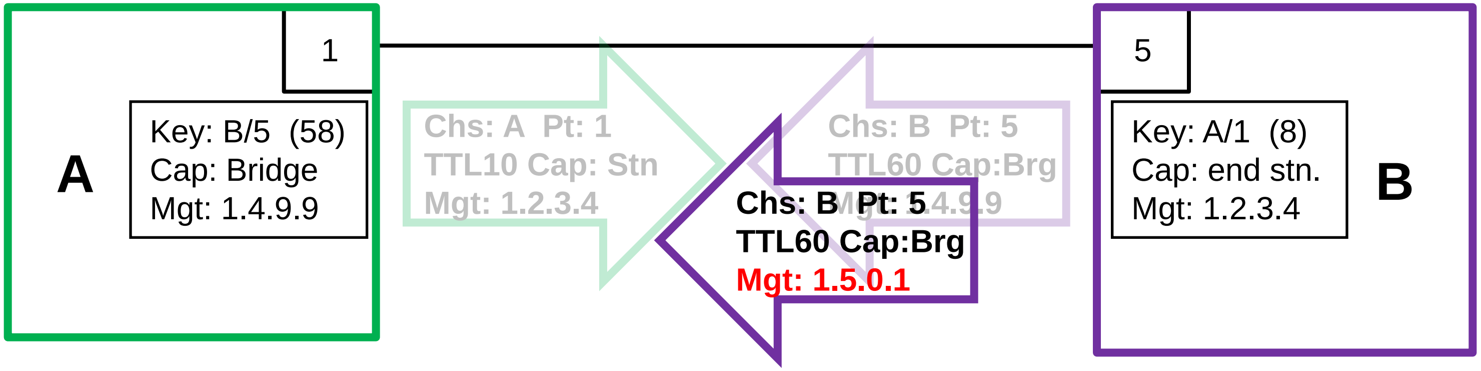
- **A says:** I am Chassis A, Port 1, this information is good for 10 seconds, I have the end station capability, and my management address is IPv4 1.2.3.4.
- **B says:** I am Chassis B, Port 5, this information is good for 60 seconds, I have the MAC Bridge capability, and my management address is IPv4 1.4.9.9.

The LLDP model of operation



- Each device records this information in a database attached to each port, accessible via an SNMP MIB (or other management vehicle) to the network manager.
- The **key** for each entry in the database is the System ID and Port ID of the neighboring system.
- The entry has a Time To Live (**TTL**) that causes the entry to be deleted when the TTL ticks down to 0.
- The **data** for each entry is all of the other TLVs in the last-received LLDPDU.

The LLDP model of operation



1

A

Key: B/5 (58)
Cap: Bridge
Mgt: 1.4.9.9

Chs: A Pt: 1
TTL10 Cap: Stn
Mgt: 1.2.3.4

Chs: B Pt: 5
TTL60 Cap:Brg
Mgt: 1.4.9.9

Chs: B Pt: 5
TTL60 Cap:Brg
Mgt: 1.5.0.1

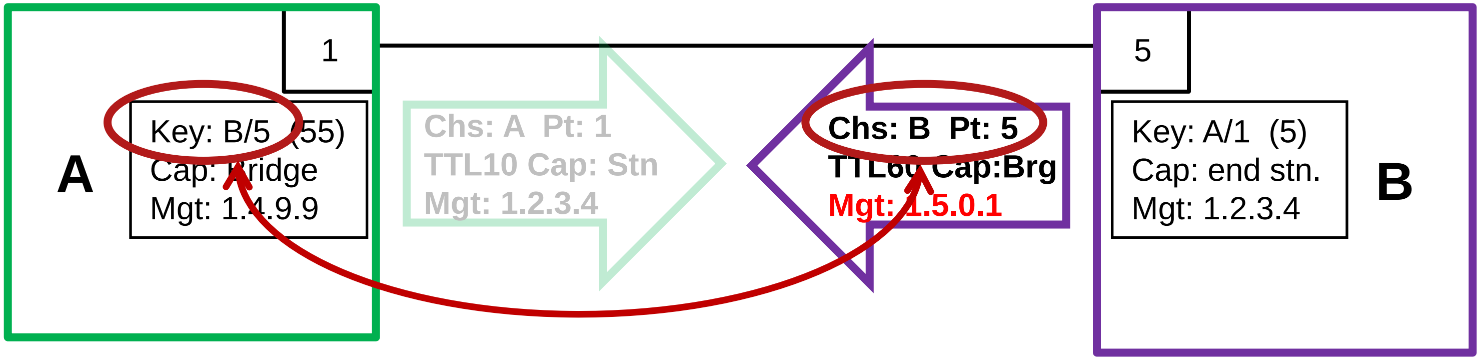
5

Key: A/1 (8)
Cap: end stn.
Mgt: 1.2.3.4

B

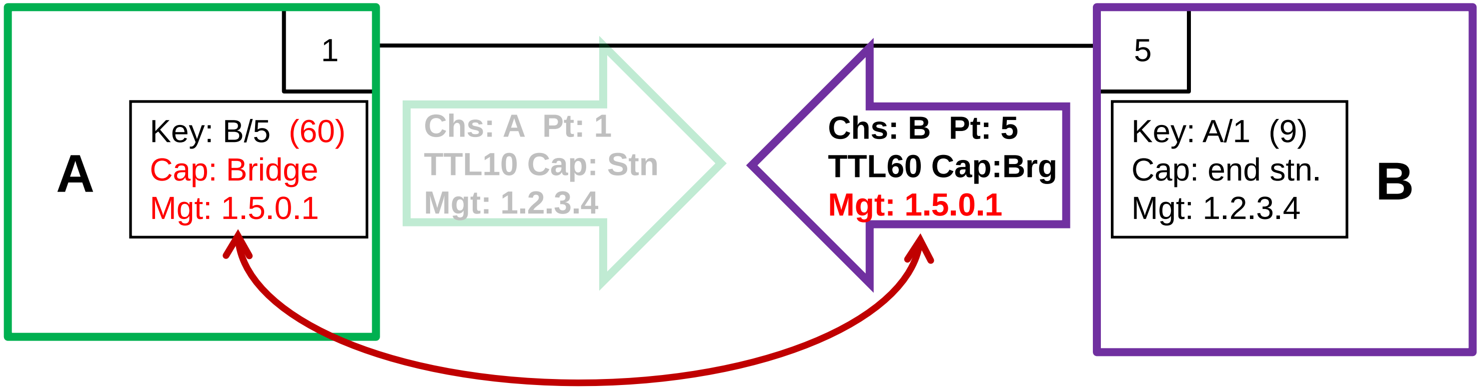
- If anything transmitted by LLDP changes (e.g., B’s management address), then LLDP re-transmits its information to update its neighbors.

The LLDP model of operation



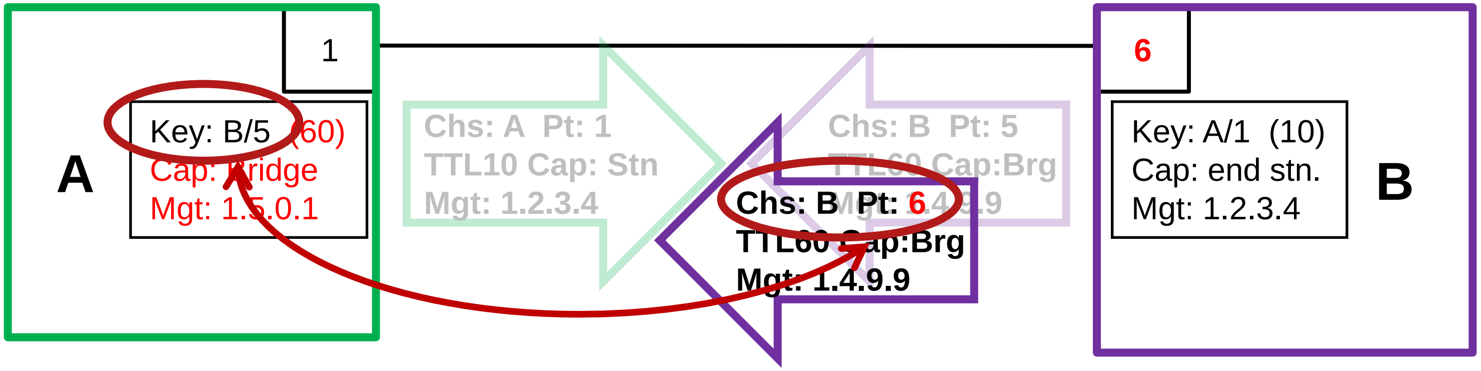
- If anything transmitted by LLDP changes (e.g., B’s management address), then LLDP re-transmits its information to update its neighbors.
- The receiver (**A**, here) **compares the key** of the LLDPDU to the keys in its database on that port.

The LLDP model of operation



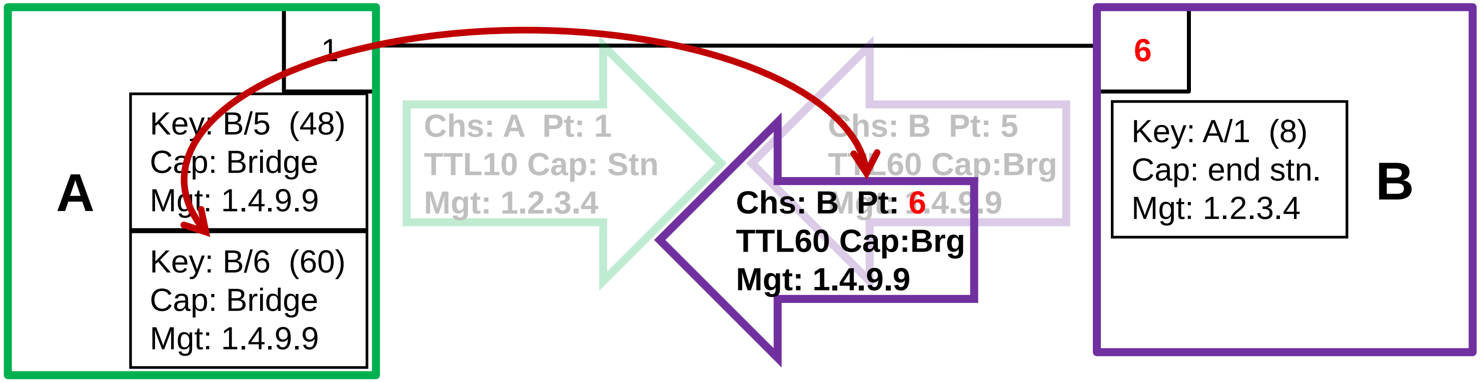
- If anything transmitted by LLDP changes (e.g., B's management address), then LLDP re-transmits its information to update its neighbors.
- The receiver (**A**, here) **compares the key** of the LLDPDU to the keys in its database on that port.
- If it finds a matching key, then the information in the LLDPDU **replaces all** of the information in the database.

The LLDP model of operation



- If the **key does not match**, (e.g., the name of the port changed), then LLDP assumes that it has found a new neighbor.

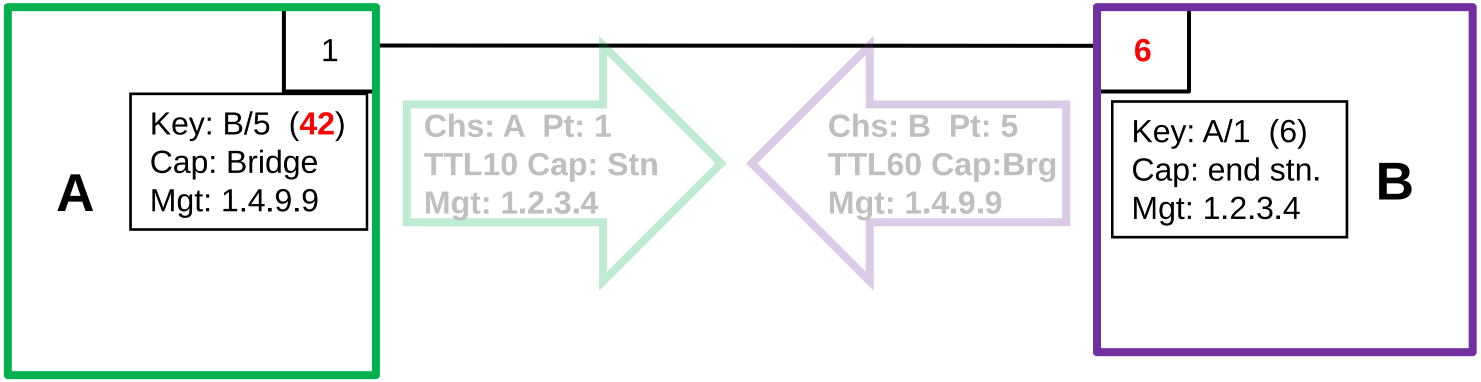
The LLDP model of operation



- If the **key does not match**, (e.g., the name of the port changed), then LLDP assumes that it has found a new neighbor.
- The **new neighbor** (new key) is added to the database.

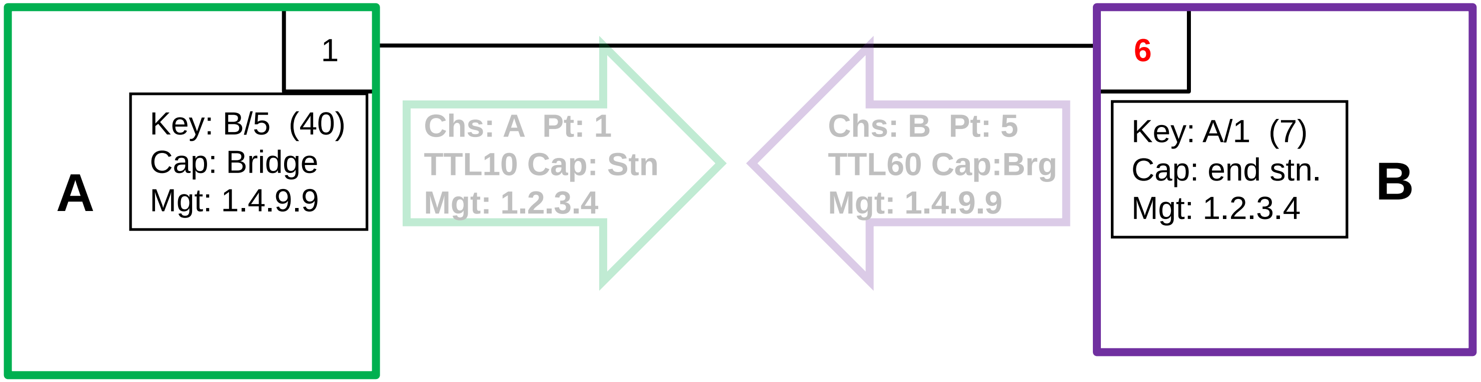
  This is the right behavior, because Chassis IDs and Port names seldom change; receiving a new Chassis/Port key normally means one has found an additional neighbor. There is a way to withdraw transmitted information (transmit TTL=0) if a system’s key information changes, or if it is about to shut down.

Transmission loss protection



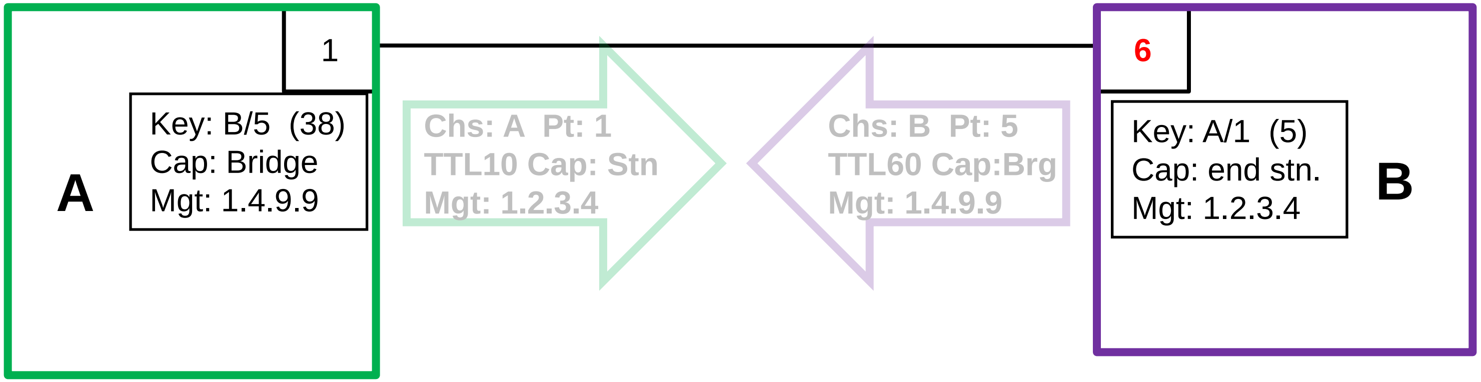
- The TTL in the database ticks down. The transmitter is expected to retransmit LLDP at a rate sufficient that, if an LLDPDU is lost in transit, another will be transmitted before the receiver's database expires.
- Normal transmission rate is one LLDPDU every 30 seconds, with a TTL value of 64 seconds, which is protects against the loss of a single LLDPDU.

Fast transmission mode



- When a link first comes up, or when any of the transmitted data changes, LLDP goes into fast transmission mode, and transmits four LLDPDUs at 1-second intervals (all with the normal 64 second TTL).
- This ensures that the neighbor sees changes to a system’s information promptly.

No Acknowledgements



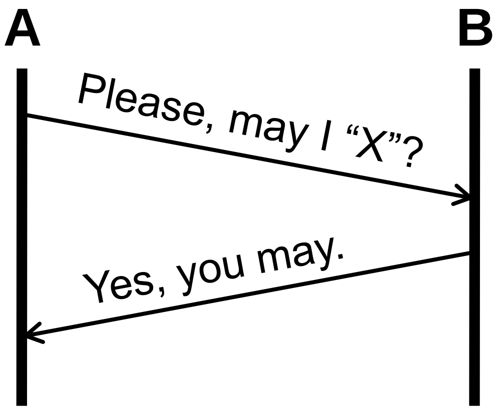
- **LLDP does not acknowledge transmissions.**
- LLDP is not a bidirectional continuity assurance protocol.
- It can be useful on links that are accidentally (or even purposely) unidirectional.

Request / response protocols

The ~~Problem~~ Opportunity

- Given that:
  - LLDP has a convenient reach.
  - LLDP is widely implemented.
  - TLVs can carry any kind of information.
  - Any organization with an OUI can define new TLVs for free.
  - Acquiring an EtherType costs money and time.
- It’s easier to develop and deploy a new protocol over LLDP than to create a new protocol from scratch using an EtherType.
- **Good news**: Using LLDP as a protocol carrier can **save** time and buffer space during the bring-up of a link.
- **Bad news:** Using LLDP as a protocol carrier can **waste** time and buffer space during the bring-up of a link.

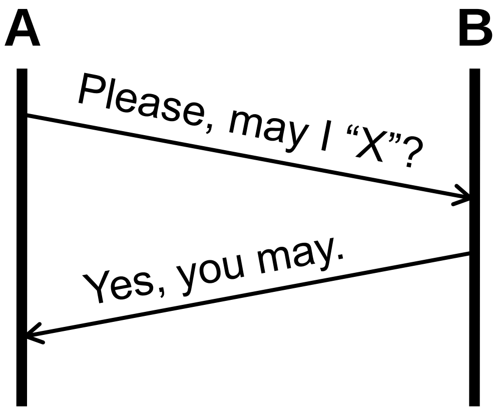
Request / response protocols



Does X

- Typical request/response protocol.
- Perhaps "X" is, "draw 20 watts via DTE Power."
- **Seems** very simple.

Request / response protocols



A

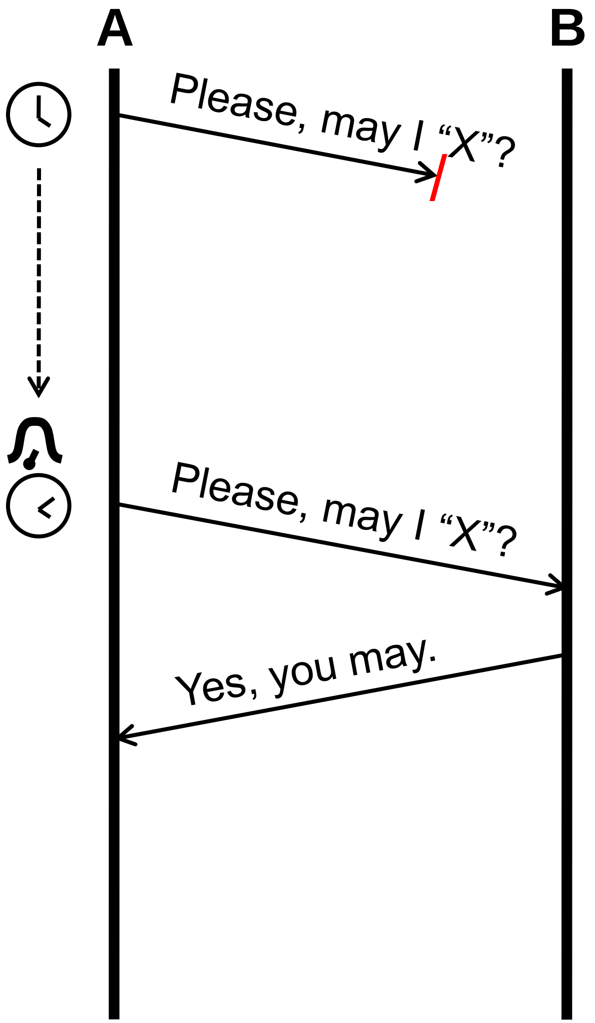
B

Please, may I "X"?

Yes, you may.

Does X

- But, what if a PDU is **lost**?
- We need a timer to resend.



A

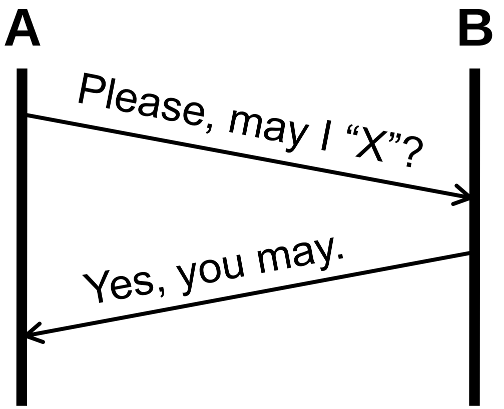
B

Please, may I "X"?

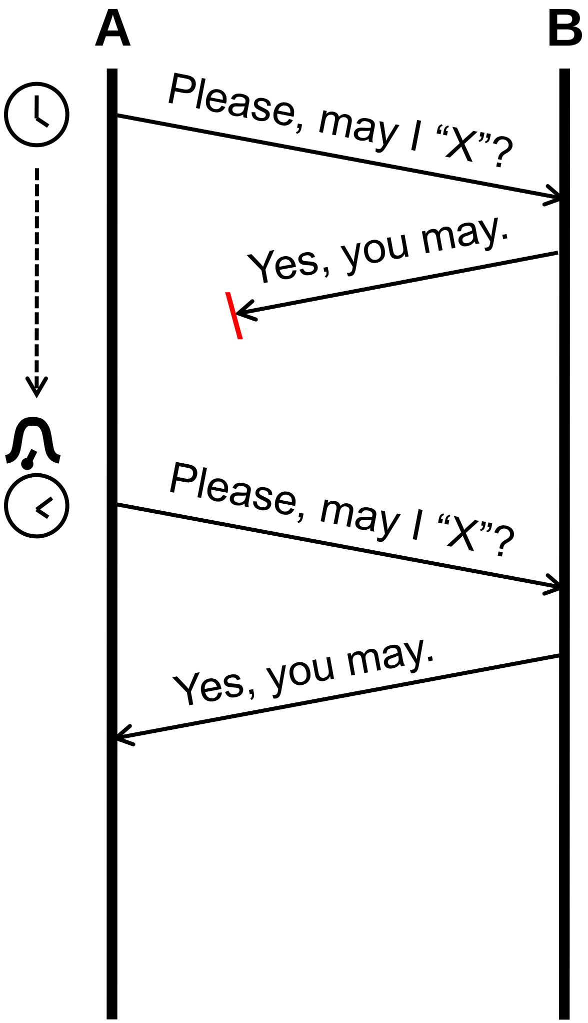
Please, may I "X"?

Yes, you may.

Request / response protocols

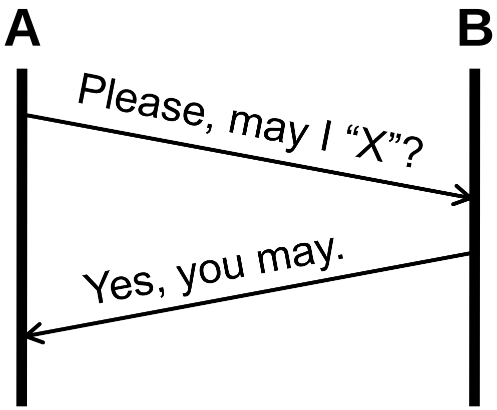


Does X

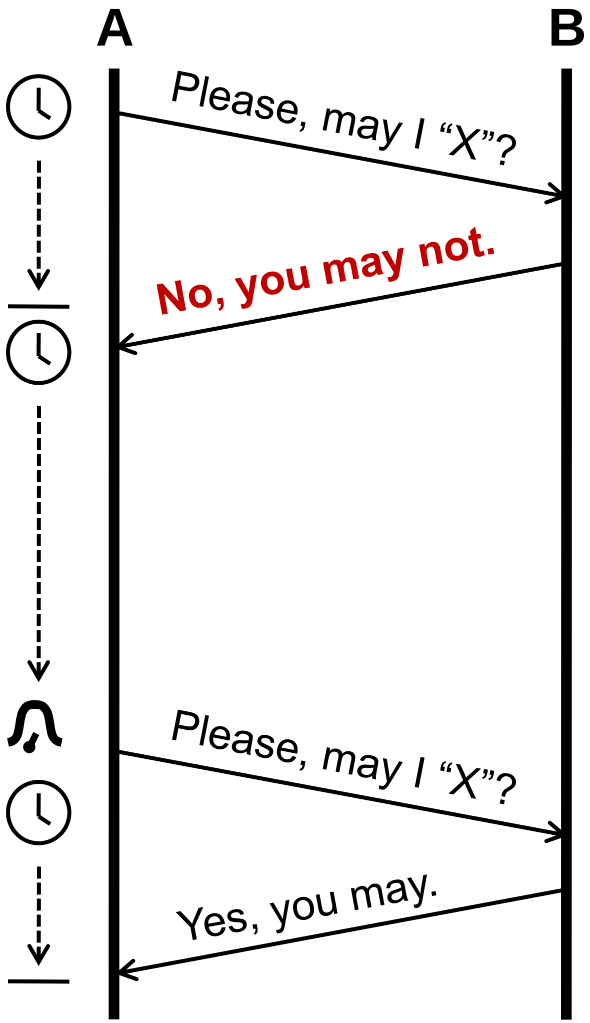


- But, what if a PDU is **lost**?
- We need a timer to resend.

Request / response protocols

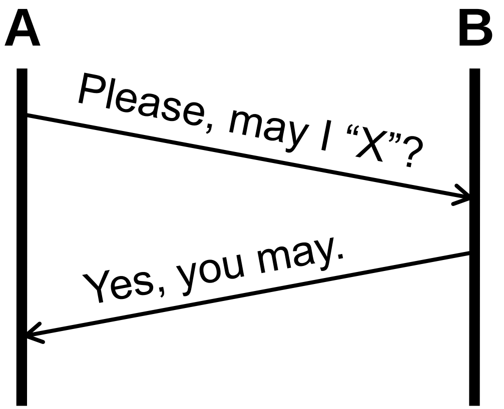


Does X

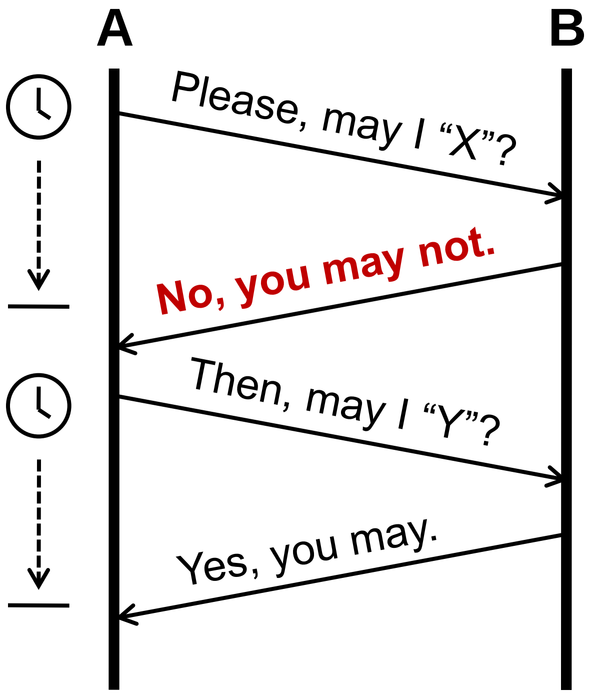


- What if the answer is, **“No.”**
- Does A ask again, later?
- How much later?

Request / response protocols



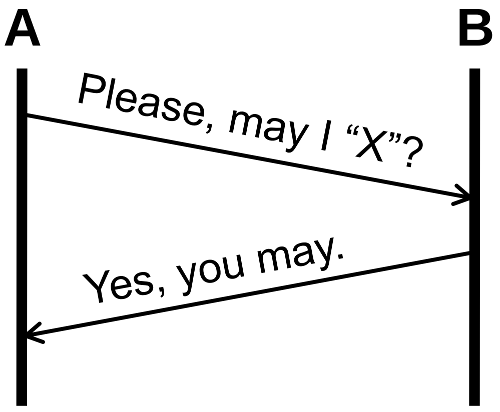
Does X



Does Y

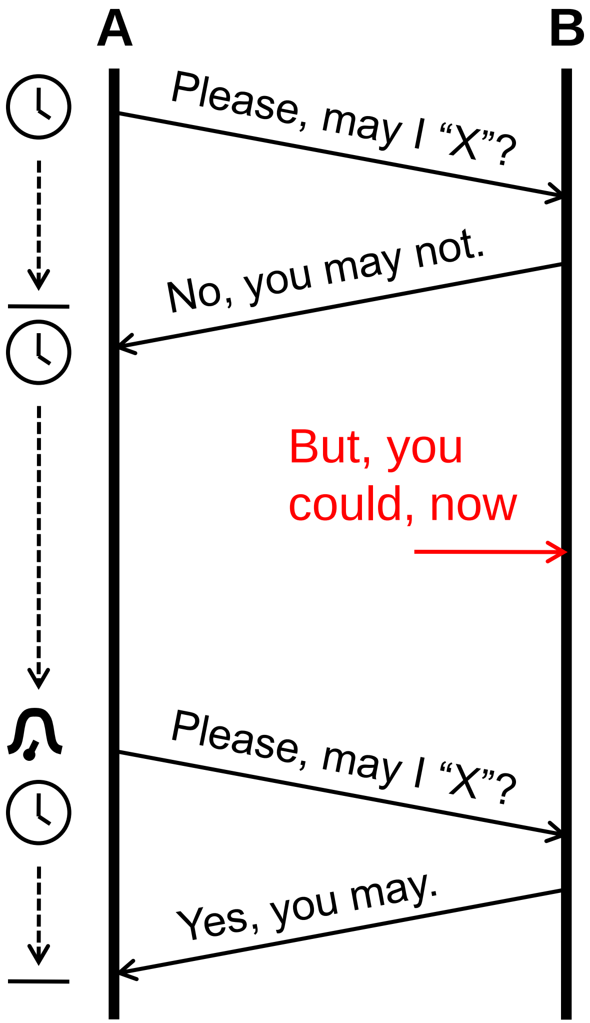
- What if the answer is, **"No."**
- Does A ask for second-best?

Request / response protocols

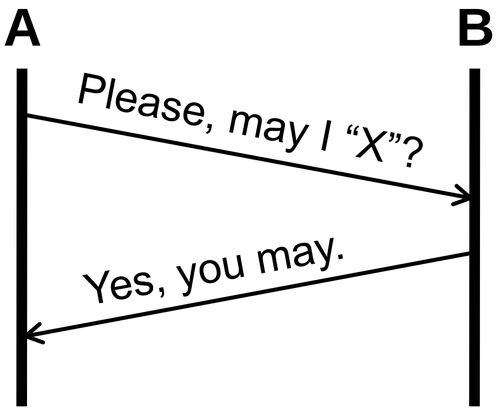


Does X

- What if conditions change?
- Do we not take advantage of the change?



Request / response protocols



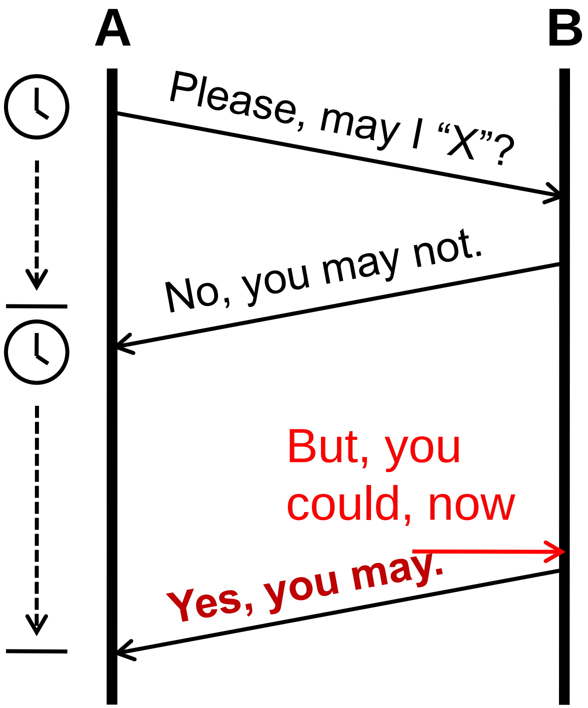
A

B

Please, may I "X"?

Yes, you may.

Does X



A

B

Please, may I "X"?

No, you may not.

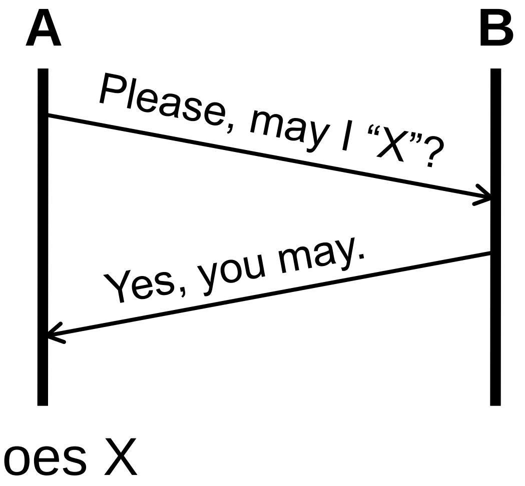
But, you could, now

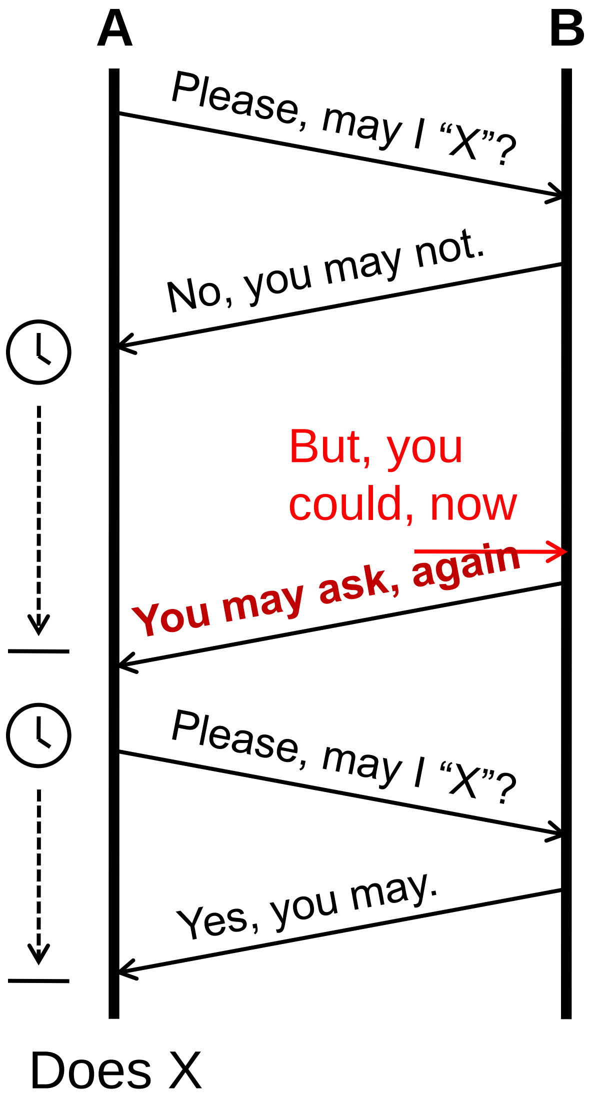
Yes, you may.

Does X

- What if conditions change?
- Do we reply at an odd time?

Request / response protocols





- What if conditions change?
- Do we add an extra message?

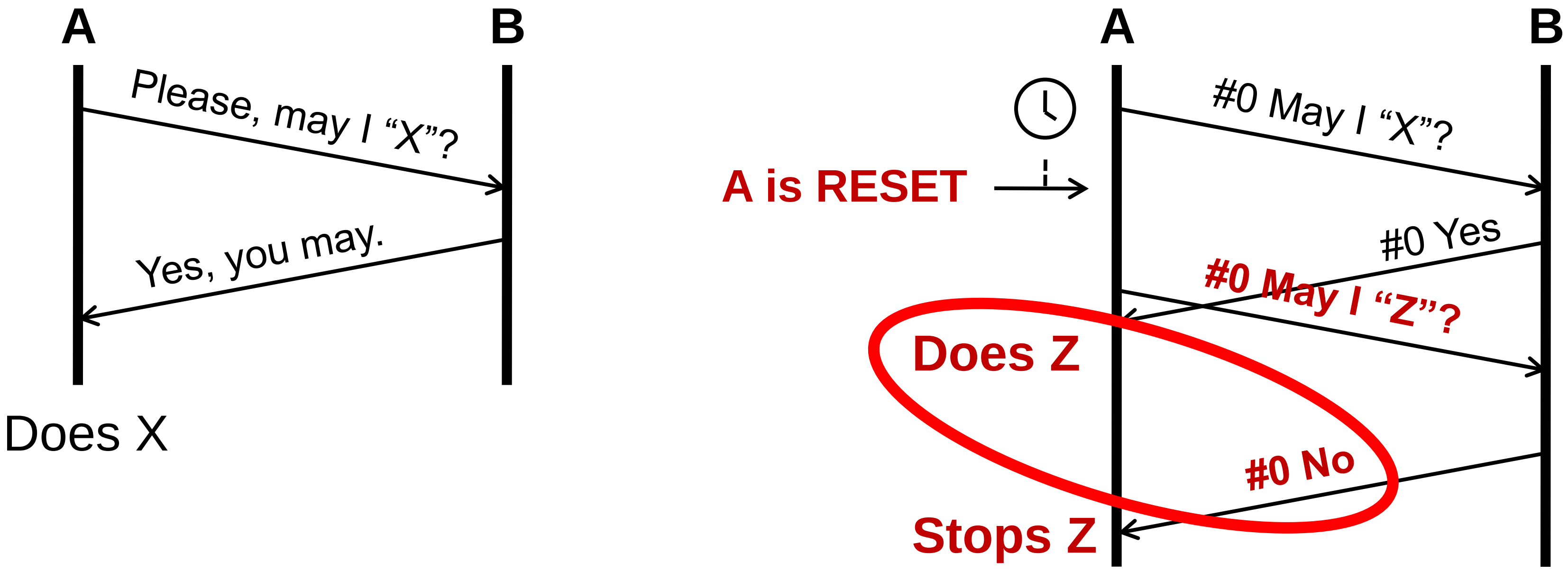
Request / response protocols

- There are clearly more questions:
  - What if my ‘Well, then, may I “Y”’ question crosses with a “Yes, you may” answer generated by a change in circumstances?
  - What if I get a “Yes, you may” and I haven’t asked for anything?
  - What if I get a “You may ask, again” and I haven’t asked for anything?
  - What if it will take me a while to figure out the answer to your question, or to arrange things to grant your request.
  - What if I change my mind about wanting something just after I send the request? Is there a “Withdraw request” request?
- The answers to these questions generally boil down to:
  - **Every request carries a request number**
  - **Every reply carries the request number triggering the reply.**
  - **Replies not matching the last-sent request are ignored.**

Request / response protocols

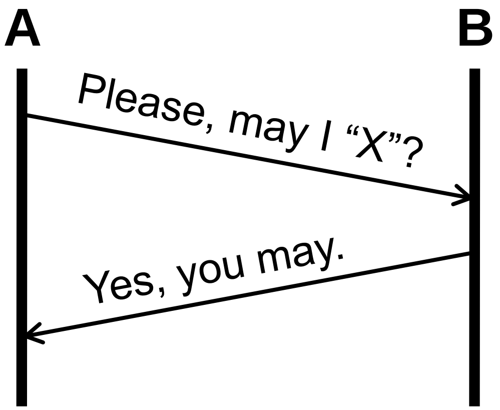
- But request numbers generate a whole new set of problems because, as the bumper sticker says, “Stuff Happens.”
- Devices can be reset.
- Operators can change configurations.

Request / response protocols



- What if requestor is reset?

Request / response protocols



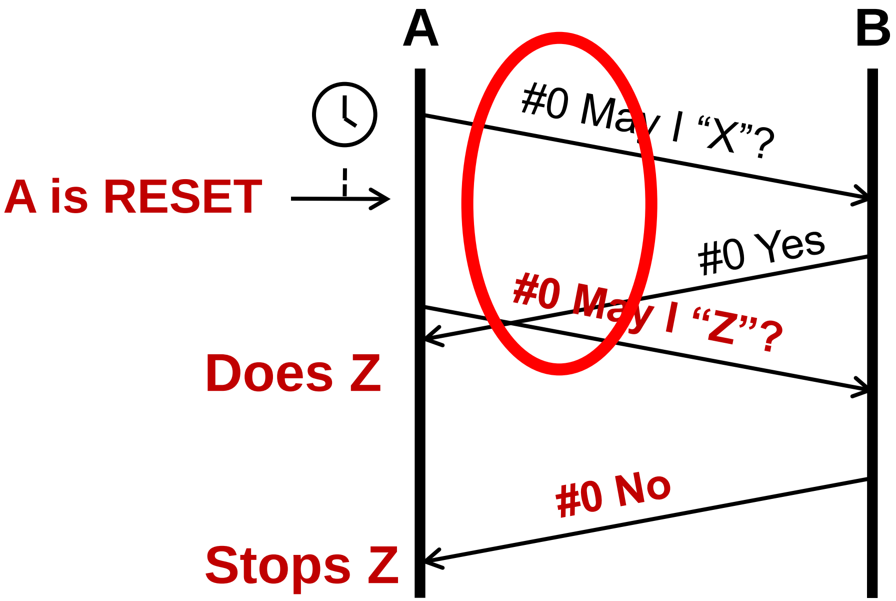
A

B

Please, may I “X”?

Yes, you may.

Does X



A

B

#0 May I “X”?

A is RESET

#0 Yes

#0 May I “Z”?

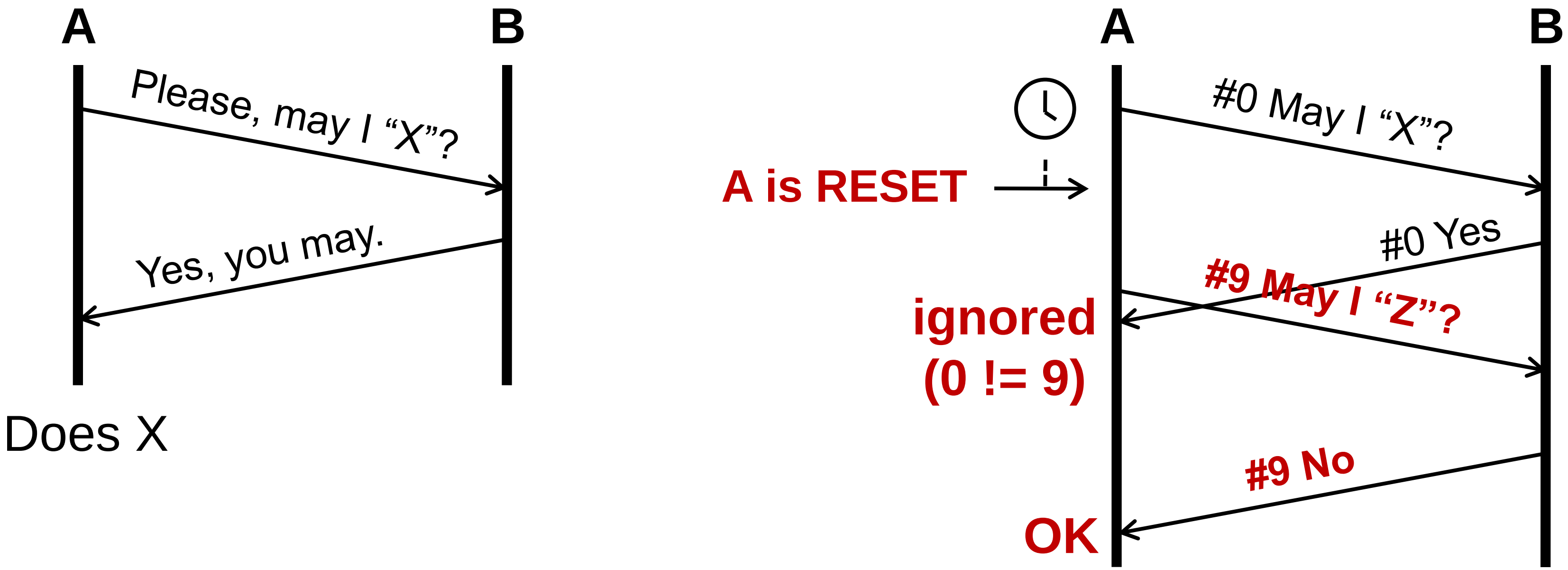
Does Z

#0 No

Stops Z

- What does B do with out-of-order requests?
- Not answering would be worse than answering.

Request / response protocols



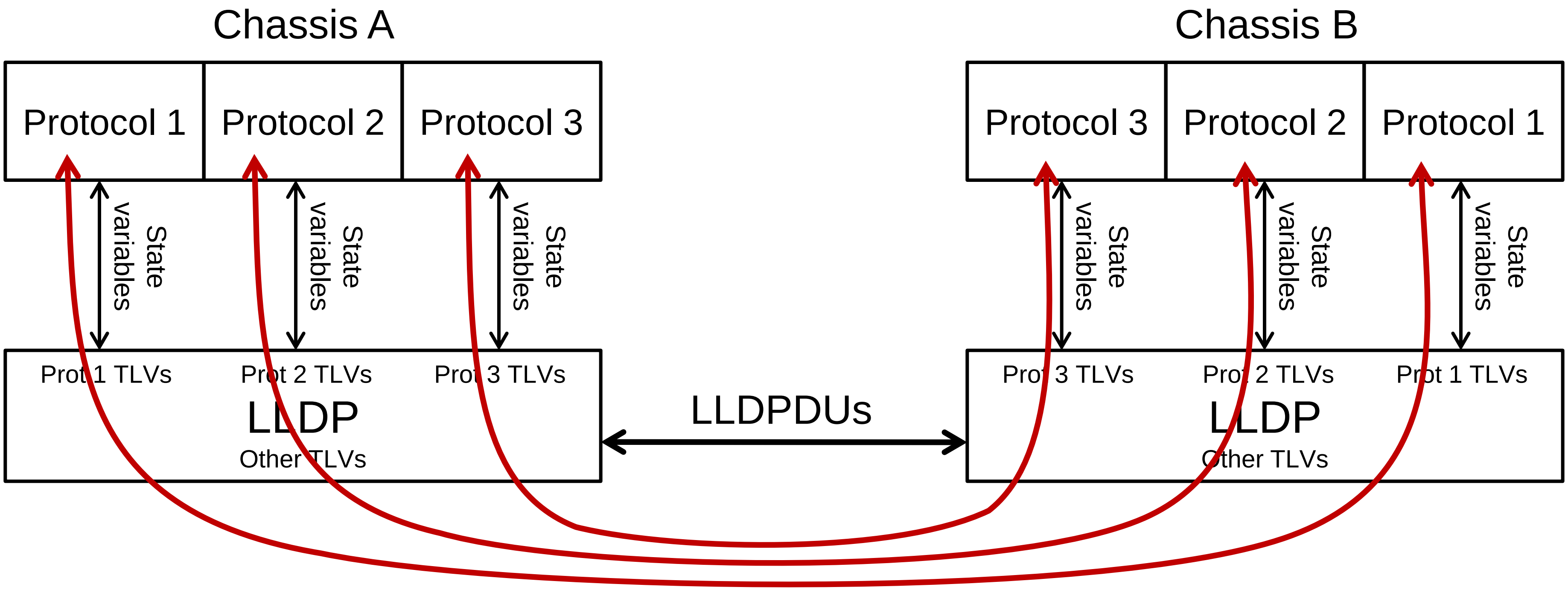
- The reset problem is usually mitigated by requiring that, when a device is reset, its sequence number is set to a random number, in order to lessen the chances of an accidental sequence number collision.

Request / response protocols

- But, the most important issue is, **“What constitutes a request?”**
- In a protocol with its own PDUs, the request is the transmission of the PDU. But, we have a carrier PDU for multiple protocols.
- We could transmit a TLV once, then immediately remove it, and pass the received TLV to the carried protocol as an event. But:
  - That violates the LLDP model of operation, which is simply to retain the last-received body of information in the database;
  - The carried protocols have to maintain their own timers, which may not be coordinated with either each other or with the existing LLDP timer, leading to extra PDU transmissions; and
  - There are other protocol paradigms that are more suitable for carriage by LLDP.

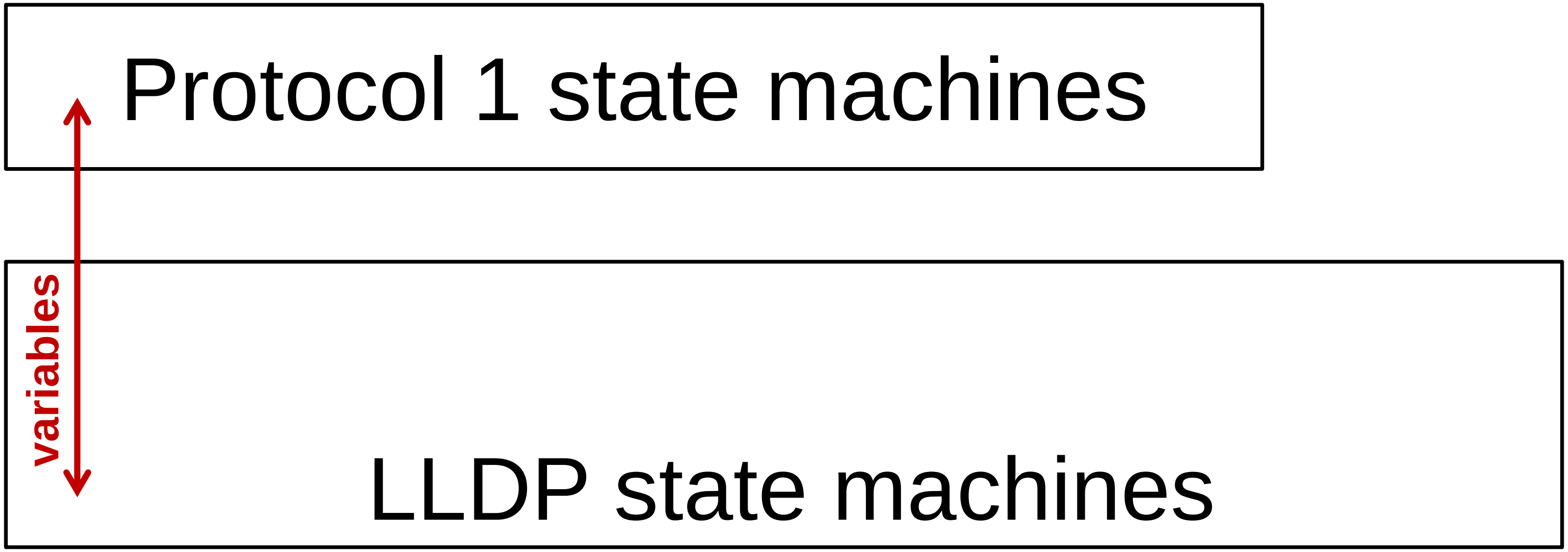
A model for LLDP as a protocol carrier

The protocol carrier model



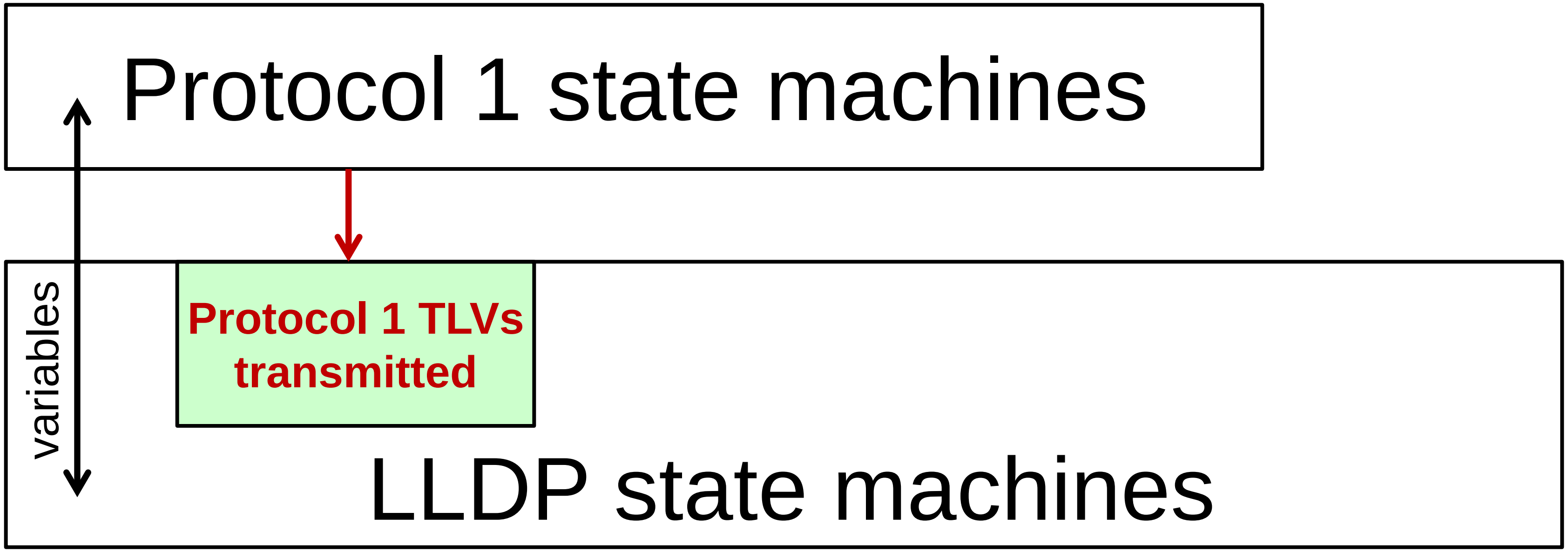
- The supported protocols peer with each other using per-protocol sets of LLDP TLVs.
- The carried protocols and LLDP communicate via a standard set of state variables.

The protocol carrier model



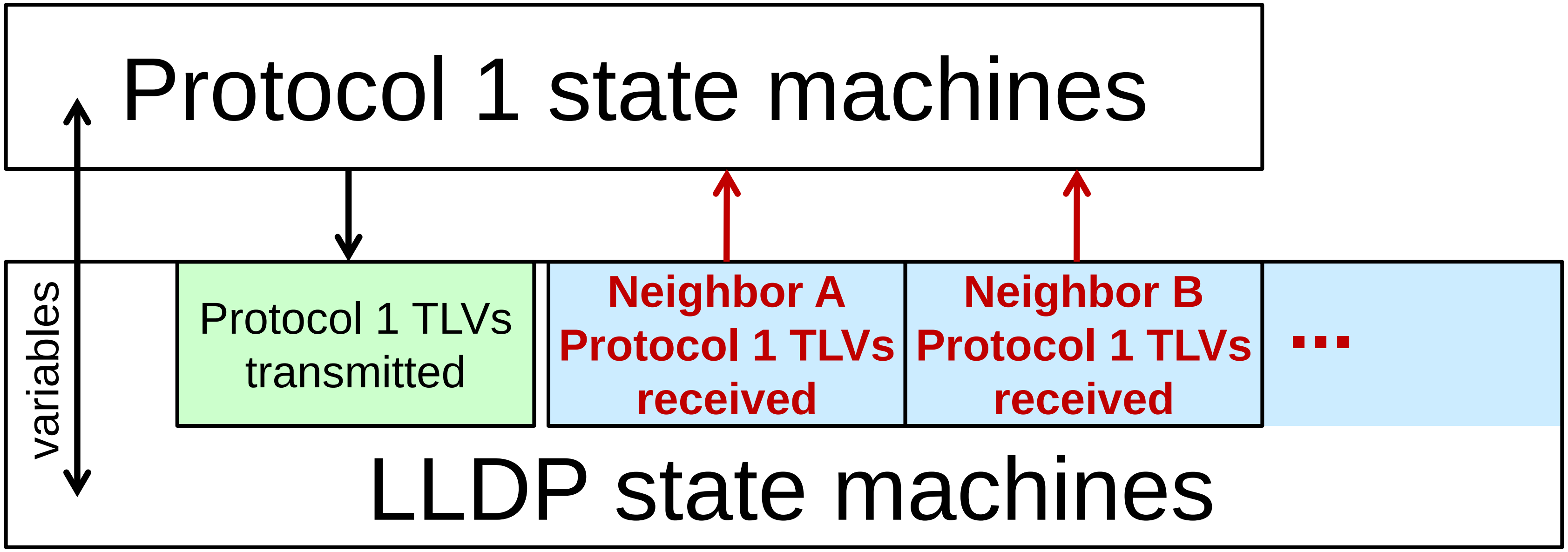
- Each protocol has its own state machines.
- LLDP has its own state machines.
- A set of standard **LLDP variables** will connect these state machines. (No special variables are needed for particular protocols.)

The protocol carrier model



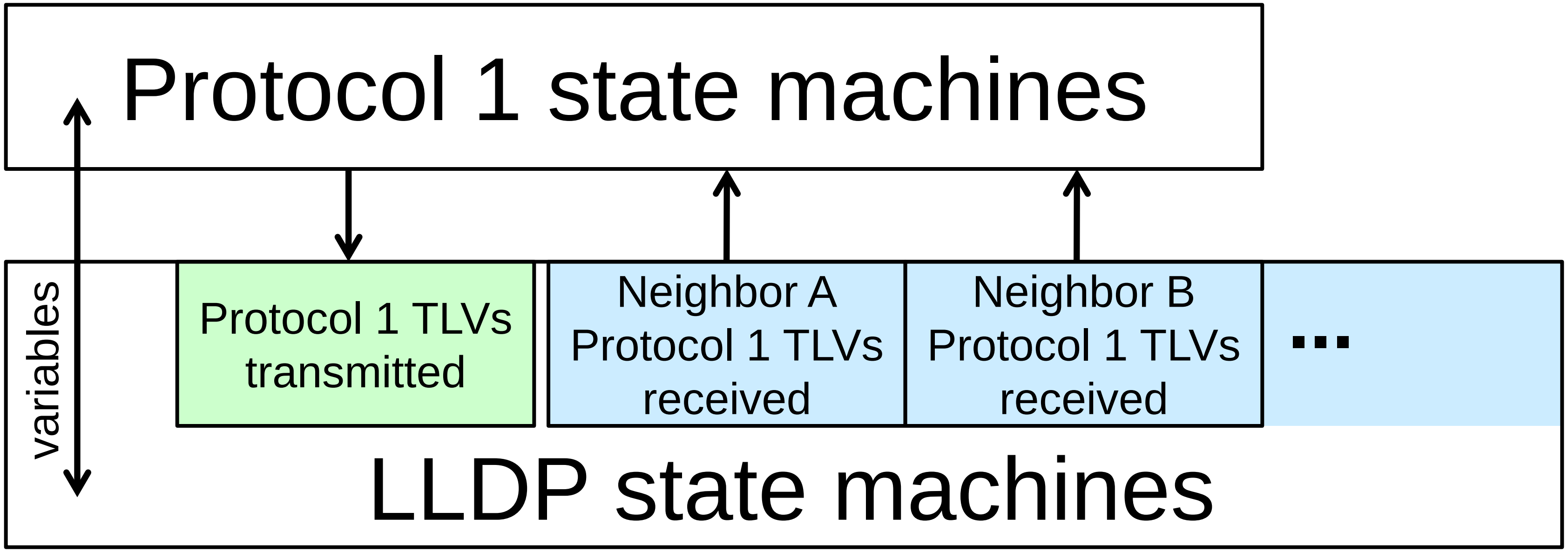
- Each protocol provides zero or more **protocol-specific TLVs** to LLDP for transmission.
- Adding, deleting, or changing the value of any of these TLVs (signaled through LLDP variables) will trigger fast transmit mode in LLDP, to ensure delivery.

The protocol carrier model


Protocol 1 state machines
variables
Protocol 1 TLVs transmitted
Neighbor A Protocol 1 TLVs received
Neighbor B Protocol 1 TLVs received
...
LLDP state machines

- Whenever a new neighbor appears, an existing neighbor disappears, or any of the **protocol-specific TLVs from a neighbor change**, the carried protocol is notified via the standard LLDP variables.

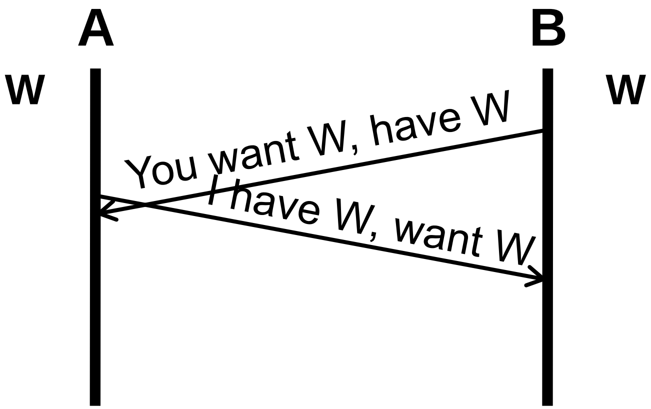
The protocol carrier model


Protocol 1 state machines
variables
Protocol 1 TLVs transmitted
Neighbor A Protocol 1 TLVs received
Neighbor B Protocol 1 TLVs received
…
LLDP state machines

- The signals are confined to TLV value changes and neighbors appearing/disappearing.
- **There are no signals** for actions or events such as the transmission or reception of an LLDPDU.

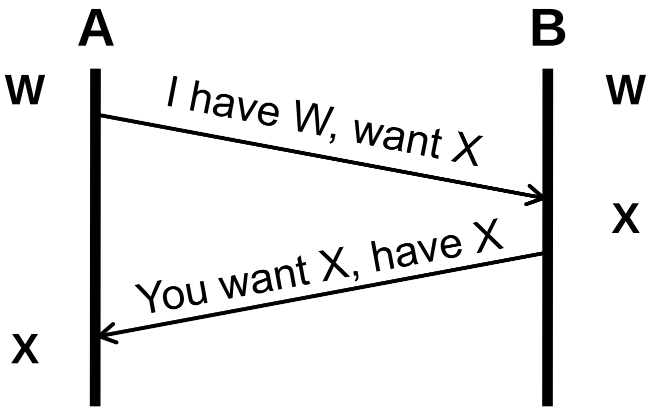
State advertisement protocols

State advertisement protocols



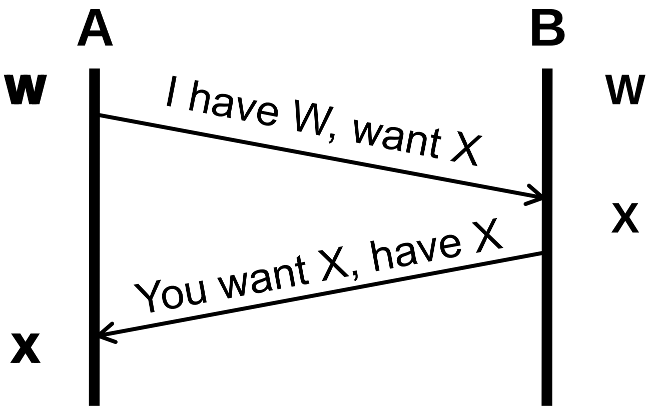
- Typical state advertisement protocol.
- “W” is the reset state, perhaps “draw 10 watts via DTE power.”
- LLDP is exchanging data, but LLDPDUs are not being transmitted in response to received LLDPDUs.

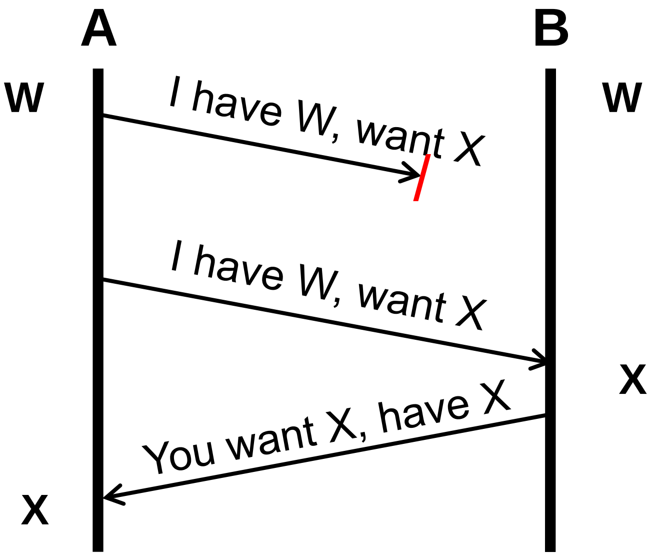
State advertisement protocols



- Typical state advertisement protocol.
- Perhaps “X” is, “draw 20 watts via DTE power.”
- B learns A’s need, changes state, advertises state.

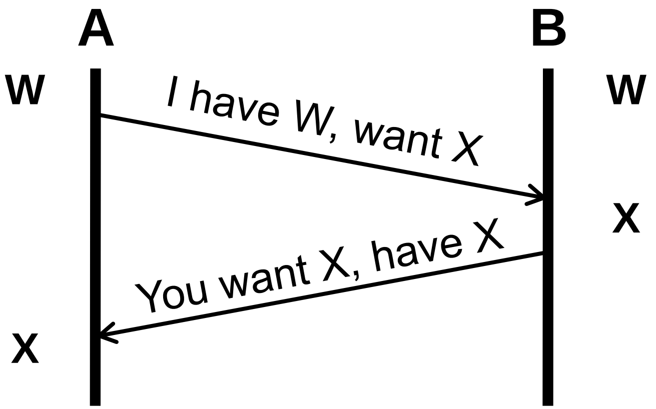
State advertisement protocols

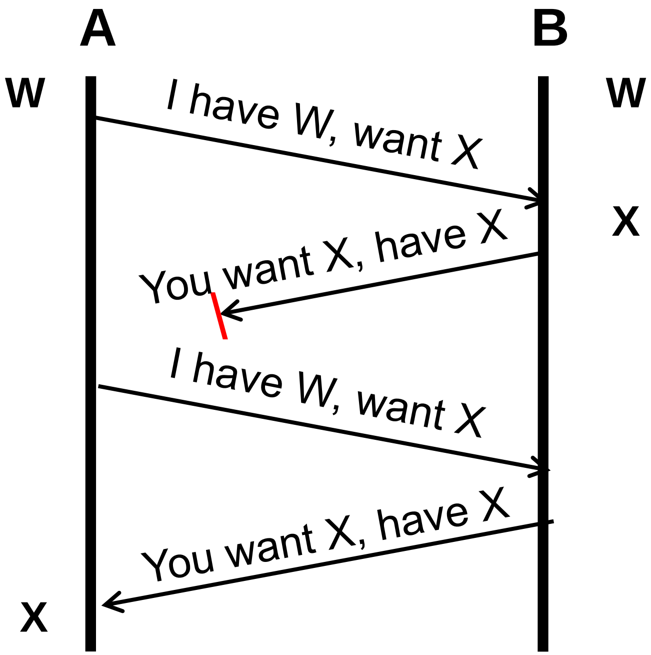




- But, what if a PDU is **lost**?
- The existing LLDP timer triggers a repeat LLDPDU, whether it was received or not.

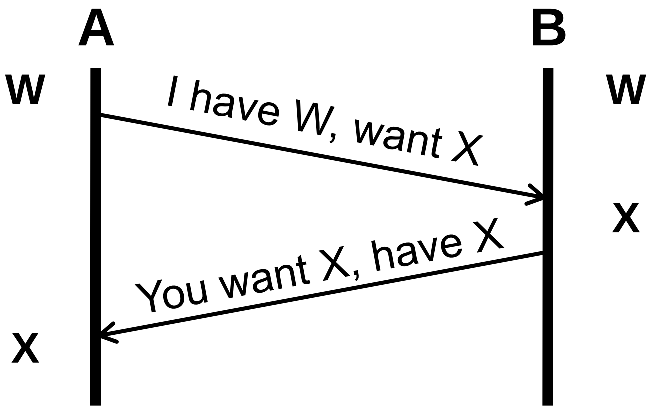
State advertisement protocols

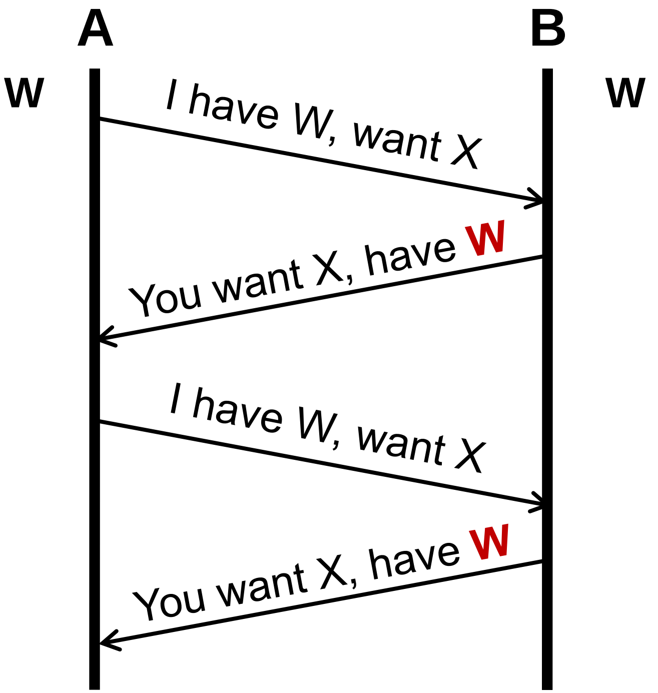




- But, what if a PDU is **lost**?
- The existing LLDP timer triggers a repeat LLDPDU, whether it was received or not.

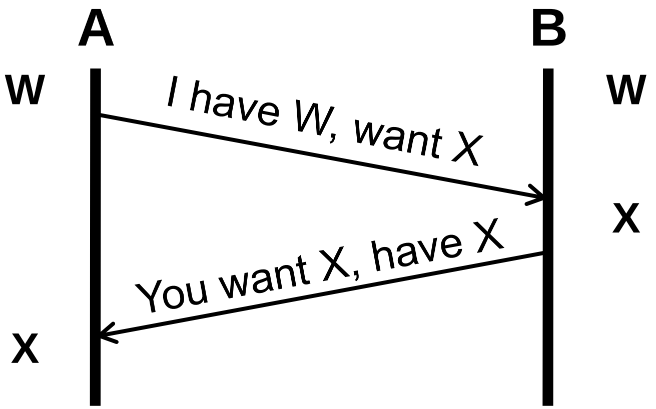
State advertisement protocols

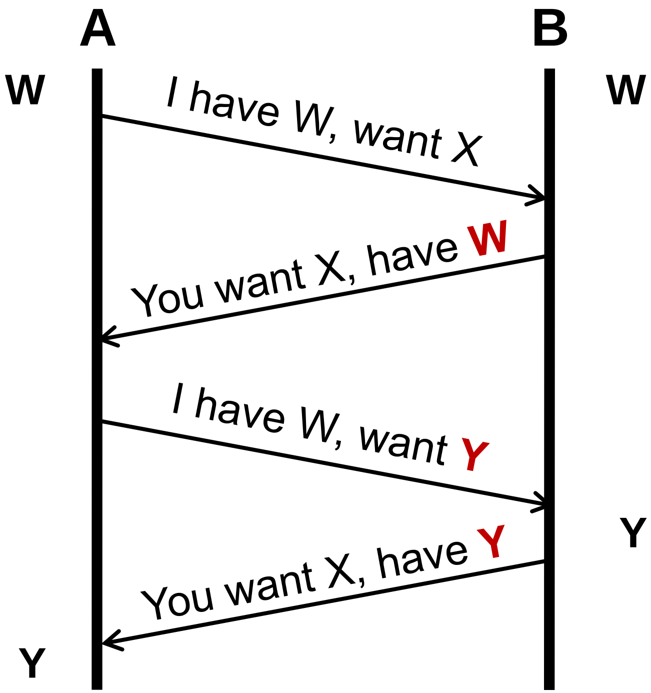




- What if the answer is, **"No."**
- A just has to wait.

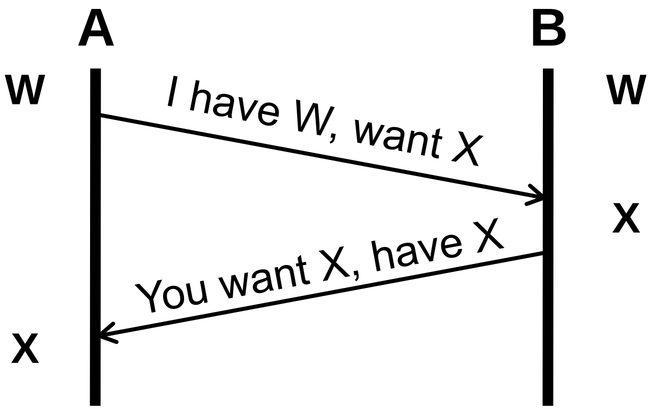
State advertisement protocols

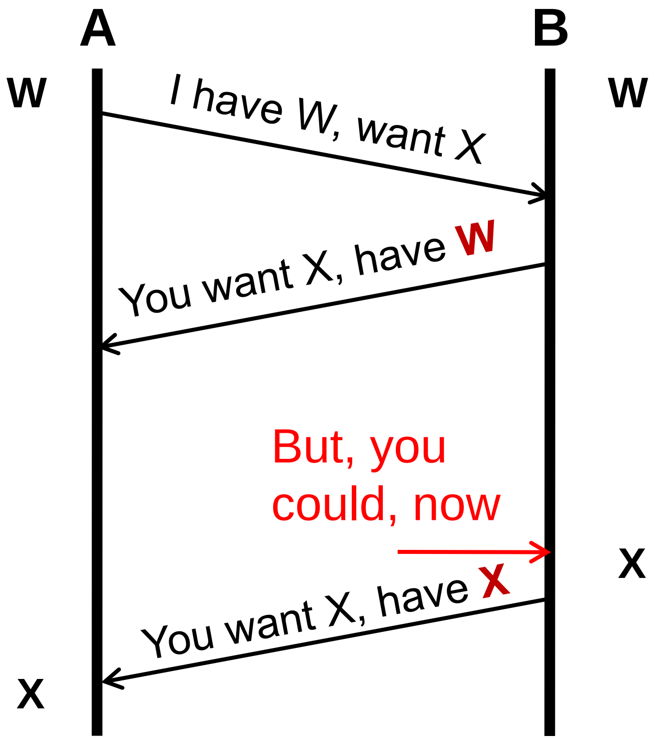




- What if the answer is, **“No.”**
- Does A ask for second-best?
- A can, but not until it receives the “You want X” from B.
- (It will get that, because LLDP retransmits.)

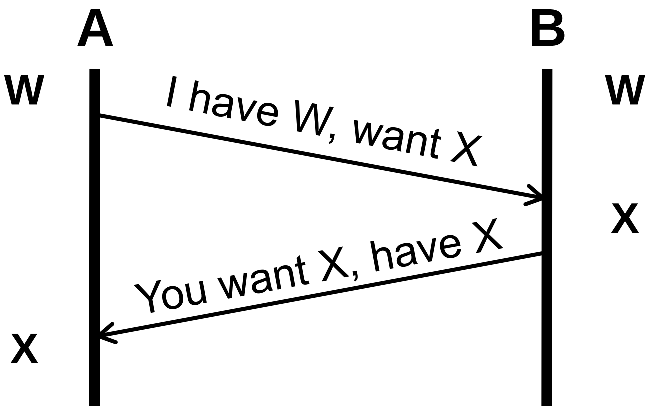
State advertisement protocols

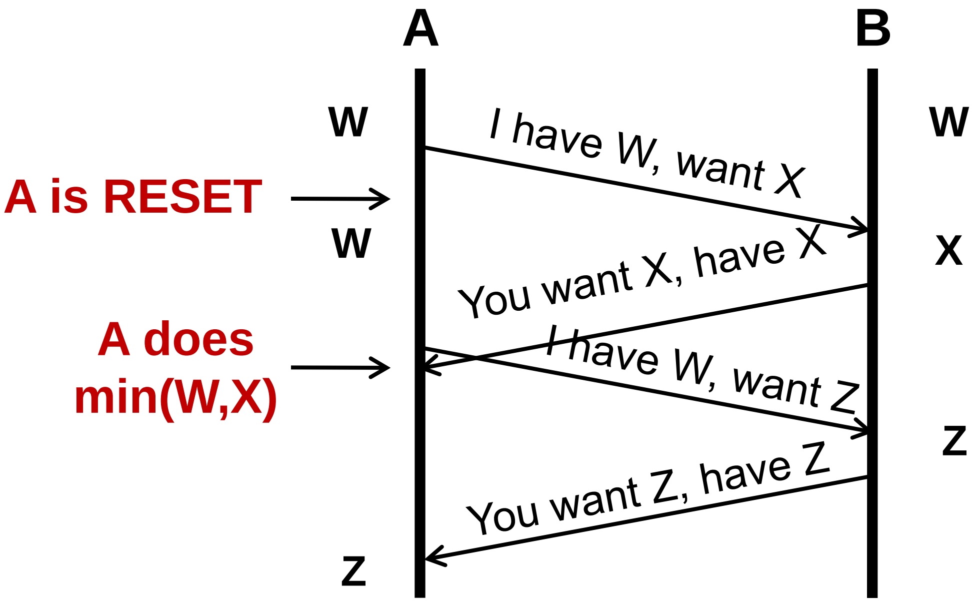




- What if conditions change?
- The new state is reported, and
  A gets the advantage immediately.

State advertisement protocols





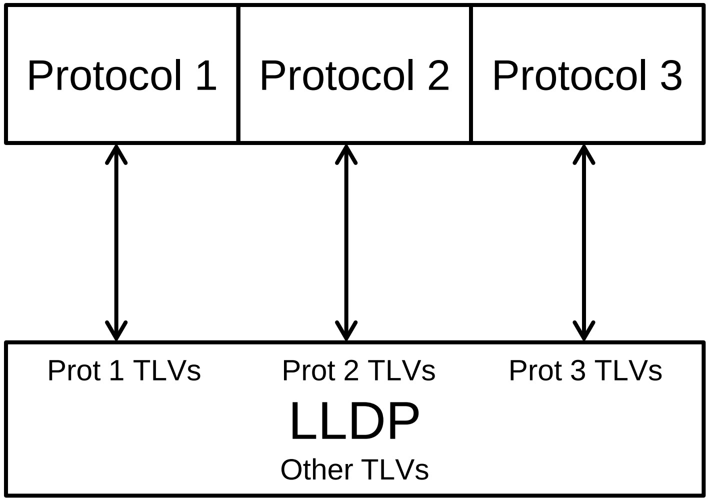
- What if requestor is reset?
- A knows about the problem as soon as it hears from B.
- On B’s side, there is no problem.
- Similar things happen if B resets.

State advertisement protocols

- Carried protocols do not need their own timers.
- Sequence numbers are not required, do not have to be initialized, compared, or incremented.
- LLDP operates in its accustomed mode.
- States **do** have to be transmitted.
  - One can think of a sequence number as an alias for the state information which is subject to getting out-of-synch with that information.

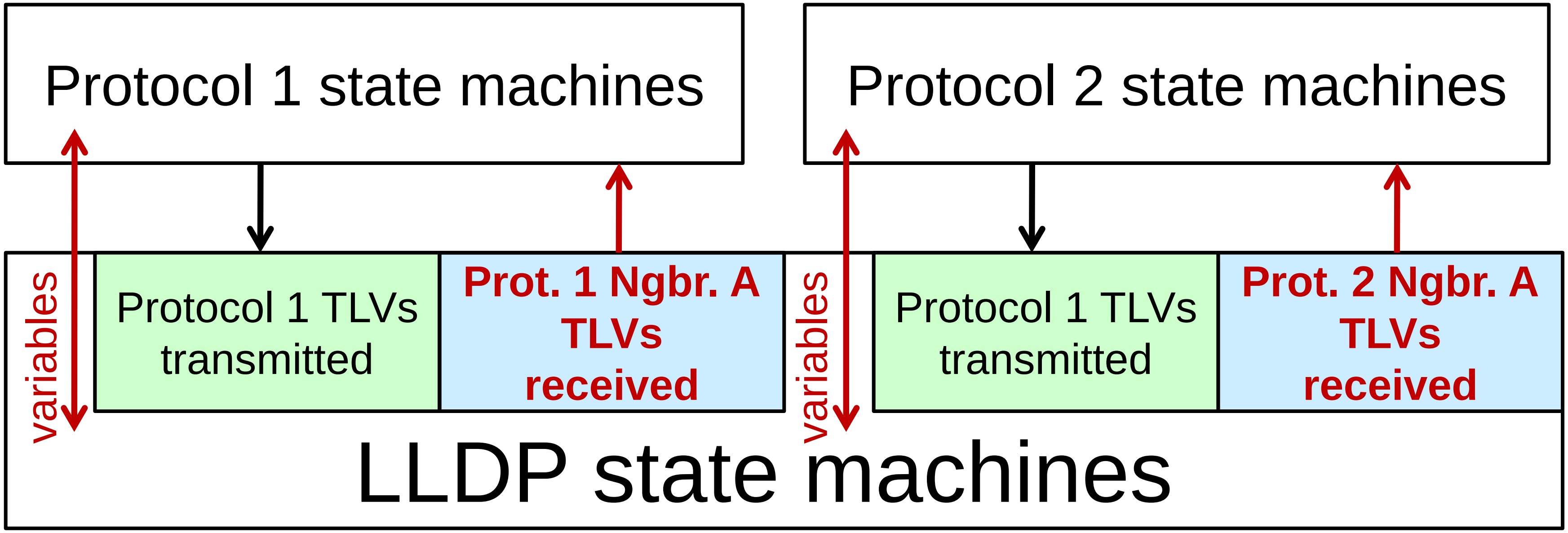
Implementation tips

Implementation tips



- If the number of LLDPDUs transmitted is the same as for the equivalent independent protocols, then transmitting all protocols’ information in each LLDPDU is a net loss.
- LLDP has a “chatty timer” limit on transmissions, which further emphasizes the need to minimize transmissions.

Implementation tips



Protocol 1 state machines

Protocol 2 state machines

variables

Protocol 1 TLVs transmitted

Prot. 1 Ngbr. A TLVs received

variables

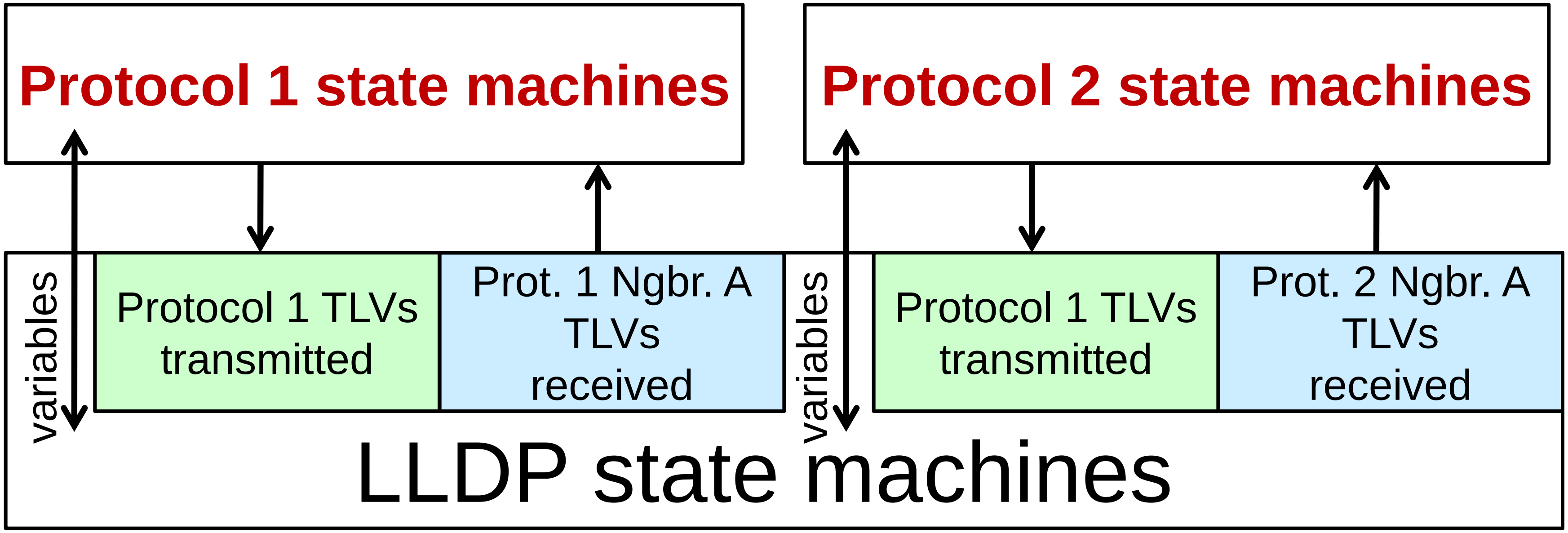
Protocol 1 TLVs transmitted

Prot. 2 Ngbr. A TLVs received

LLDP state machines

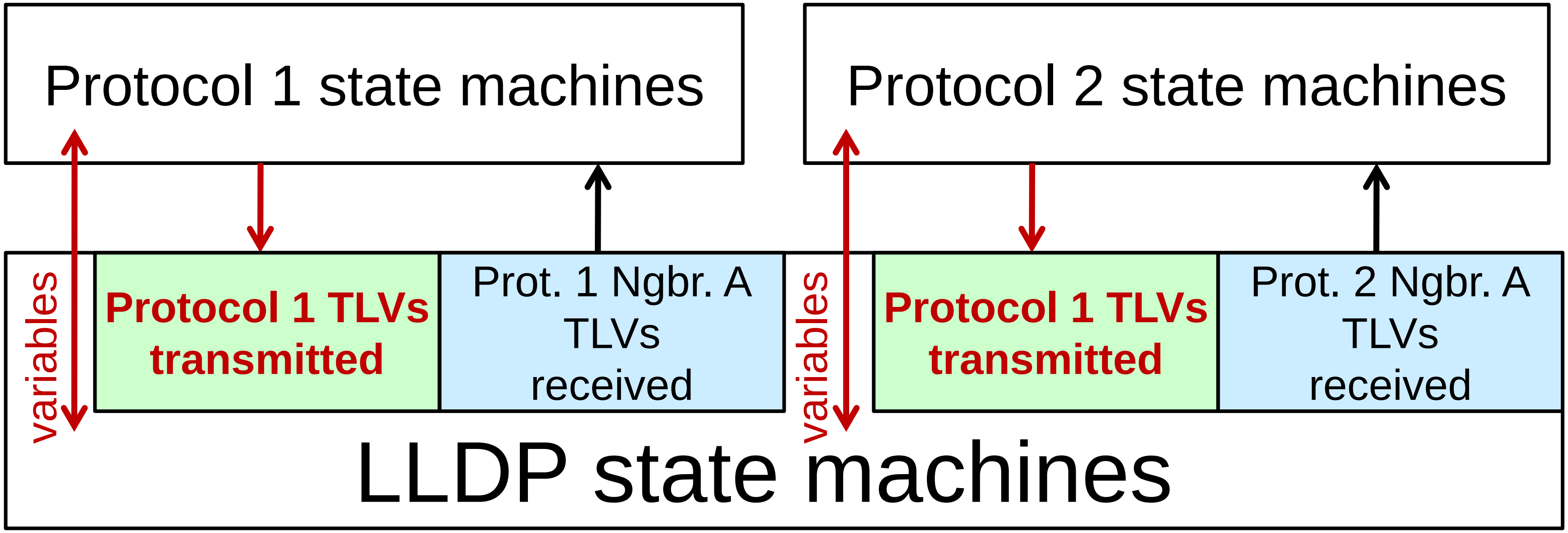
- When a **new LLDPDU** is received, all protocols whose TLVs change are notified of the change in TLV values. (Note: change in TLV values – not LLDPDU arrival.)

Implementation tips


Protocol 1 state machines
Protocol 2 state machines
variables
Protocol 1 TLVs transmitted
Prot. 1 Ngbr. A TLVs received
variables
Protocol 1 TLVs transmitted
Prot. 2 Ngbr. A TLVs received
LLDP state machines

- The Protocol state machines operate in parallel, generating their responses, if any.

Implementation tips


Protocol 1 state machines
Protocol 2 state machines
variables
Protocol 1 TLVs transmitted
Prot. 1 Ngbr. A TLVs received
variables
Protocol 1 TLVs transmitted
Prot. 2 Ngbr. A TLVs received
LLDP state machines

- After all of the protocols have supplied their new values, LLDP transmits all of the updates in one LLDPDU.

Implementation tips

- Within limits, a protocol should be allowed to effect changes such as hardware reconfiguration before returning its new TLV values.
  - If such changes take too long, then other protocols will be slowed down.
- Events outside LLDP can trigger the need to change the TLVs sent by a protocol, but this seems inevitable.