Why a standard link layer solution is needed

Paul Congdon
Hewlett Packard Company
paul_congdon@hp.com

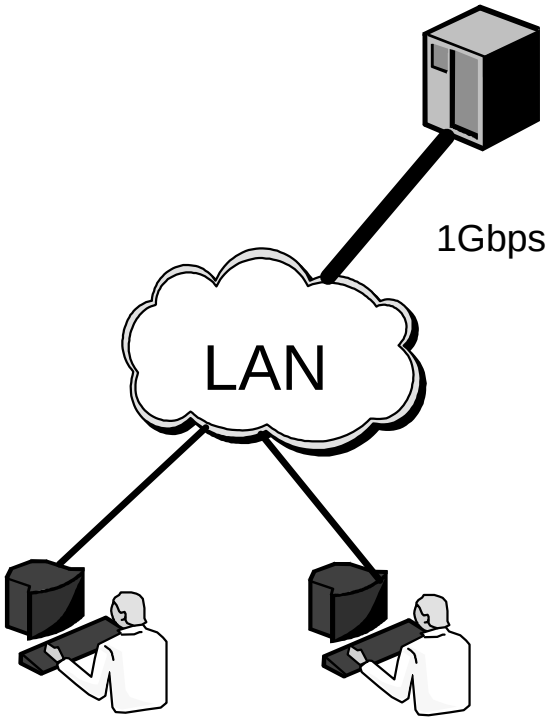
November 11, 1997
IEEE 802 Tutorial - Montreal

Port Trunking for Server Access

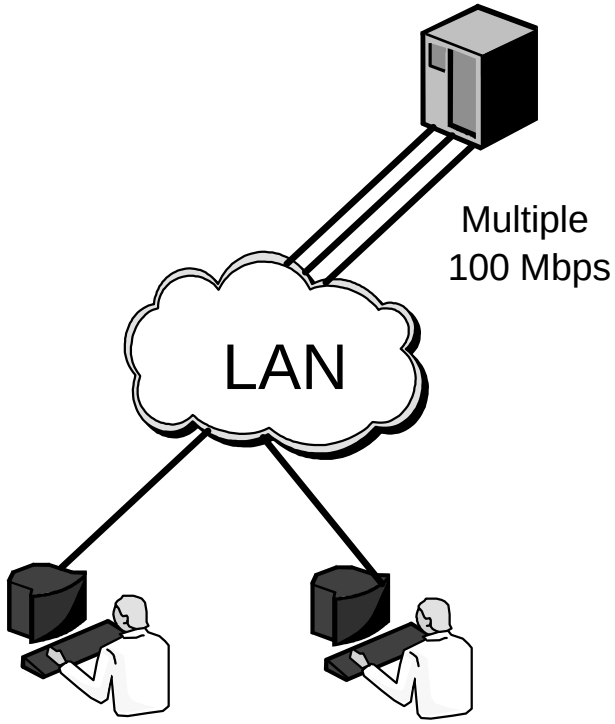
HEWLETT PACKARD

Boosting Server Access

- Move to the next higher speed



- Trunk together multiple NIC cards

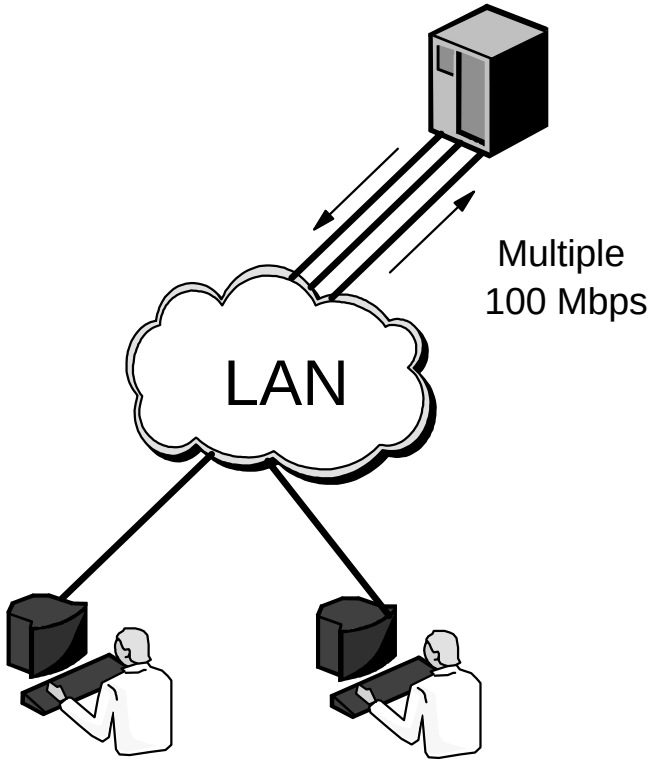


Why not **just** a higher speed link?

- Currently, it may be more cost effective to trunk multiple lower speed links
  - ✓ End-systems today may not be able to fully utilize 1Gbps
  - ✓ Available copper links and supported distances are more pervasive for lower speed links
  - ✓ 100Mbps NICs and switch ports are pretty darn cheap
- Protect investment in existing infrastructure
- Multiple links provide higher availability and resiliency
- Trunking techniques will work with future higher speed links as well

Making Server Access Transparent, Available and Fast

- Would like a single network presence
  - ✓Minimize impact of multiple links in a single system
  - ✓Provide transparency above address mapping layers
- Would like automatic link failover
  - ✓Keep servers up and running via redundancy
  - ✓Provide client transparency from Server link failures
- Would like active load balancing
  - ✓Full utilization of invested resources
  - ✓Maximize available performance



Why solve this problem at the link layer?

- Layer 1 solutions require new PHYs and MACs
- Layer 3 solutions are not as transparent to end-stations and switches
- Layer 4 solutions require application or middleware awareness

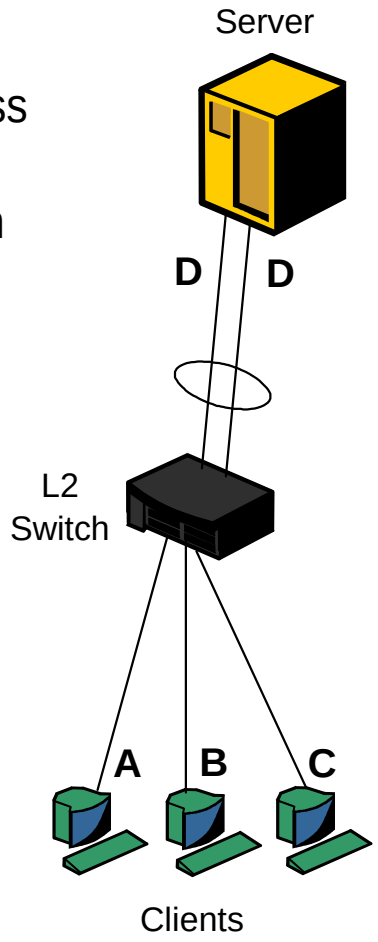
Layer 2 solution will be easy to achieve and provides the best transparency!

Solutions for Trunking at Layer 1

- Bond links at the Physical Layer
  - ✓Split packets into small fragments, transmit with loose synchronization requirements and reassemble on other side.
  - ✓Example: ISDN Bonding
- Single Network Presence
  - ✓Yes, but at the cost of new MACs and PHYs
- Failover
  - ✓Yes, but with required synchronization protocols
- Load Balancing
  - ✓Yes, but with complex fragmentation/reassembly

Solutions for Trunking at Layer 2

- Coordinate with Switch the usage of a common MAC address on multiple links
  - ✓Use a deterministic algorithm for distributing individual PDUs across multiple links.
  - ✓Assure packets are not reordered for a particular source/destination conversation
  - ✓Example: Sun Trunking 1.0, Cisco Fast Ether Channel
- Single Network Presence
  - ✓Yes, in both the address resolution and network stacks
- Failover
  - ✓Yes, via existing link error detection
- Load Balancing
  - ✓Yes, if switch distributes using source address and server using destination



Solutions for Trunking at Layer 3

- Keep the Network Address consistent - many solutions
  - ✓ Map single Network Address to multiple MAC Addresses via special load-balance resolution protocol
  - ✓ Perform one-way load balancing by using a different MAC address on traffic leaving the end-station
  - ✓ Always route directly to the end-station
  - ✓ Example: Balance.NLM
- Single Network Presence
  - ✓ Only at network layer. MAC-to-Network Address mapping may vary
- Failover
  - ✓ Yes, if dynamic MAC-to-Network Addressing works
- Load Balancing
  - ✓ Yes, with changes to the network stack or only in one direction

Solutions for Trunking at Layer 4 and above

- Keep the Network Name consistent
  - ✓ Map single Network Name to multiple Network Addresses via special name resolution protocol
  - ✓ Remap Network Addresses for specific network connections
  - ✓ Examples: NATs, LocalDirector, Web Server Director, CORBA
- Single Network Presence
  - ✓ Yes, to the network, but not the end-stations without modification or specialization
- Failover
  - ✓ Yes, at the system level
- Load Balancing
  - ✓ Yes, with application awareness or specialized devices

Conclusions

- Port Trunking at Layer 2 is an appropriate technology to boost performance and availability of Network Server Access
- Layer 2 Trunking is equally applicable and valuable for Switch-to-Switch links
- Standardization and interoperability should be relatively straight forward

Lets begin working on a standard specification ***now!***