

Review and Update of Carrier Extension Proposal

**Howard M. Frazier, Jr.
Sun Microsystems Computer Company
Internet and Networking Products Group
9-Sept-1996
IEEE 802.3z TF**

Outline

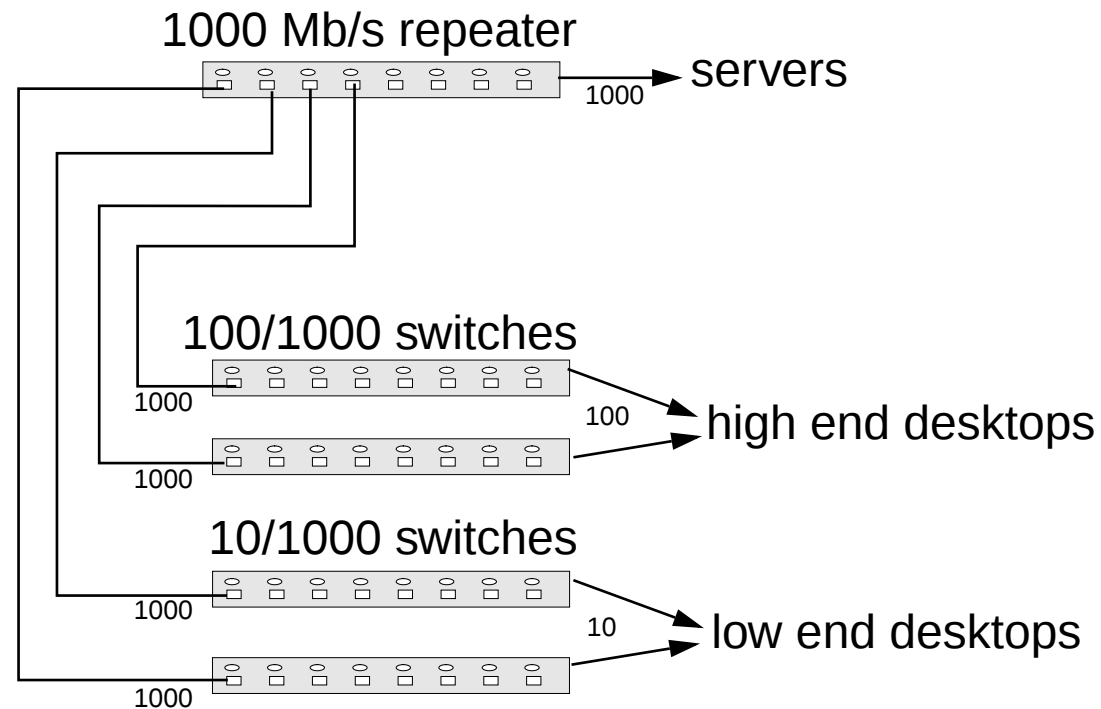
- Introduction
- Topologies
- Bit Budget
- Issues
- Solution
- Impact on MAC
- MAC Parameters
- Impact on Reconciliation Sublayer
- Performance
- Conclusions

Introduction

- **Scaling CSMA/CD to 1000 Mb/s is a good idea**
 - Widely implemented, well understood protocol
 - Demonstrated low cost at 10 and 100 Mb/s
 - Cost benefit of 1000 Mb/s shared versus 1000 Mb/s switched
- **Scaling CSMA/CD to 1000 Mb/s is slightly more complicated than scaling it to 100 Mb/s**
 - Wire delays are 10x larger (in BT) than at 100 Mb/s
 - “shift the decimal point” approach results in a ridiculously small collision domain diameter
- **Backwards compatibility with 10 and 100 Mb/s is essential**

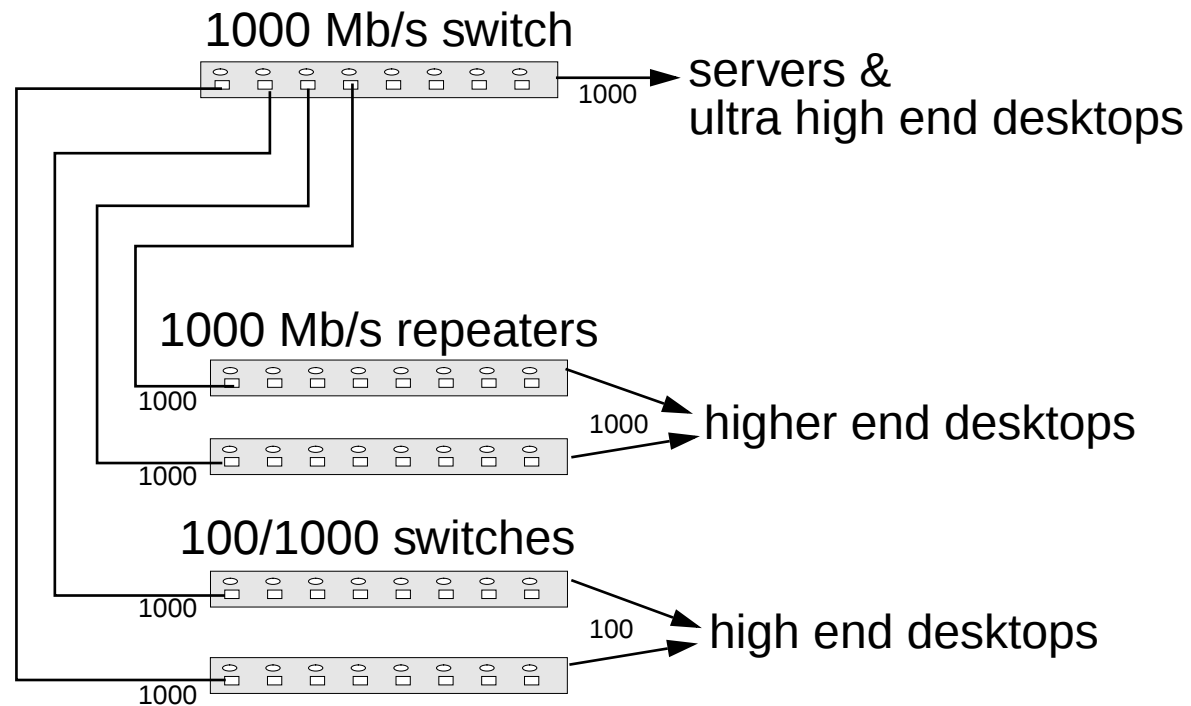
Topologies

- Most useful CSMA/CD topologies have at least one repeater



CSMA/CD used in a single repeater collision domain
within a server room

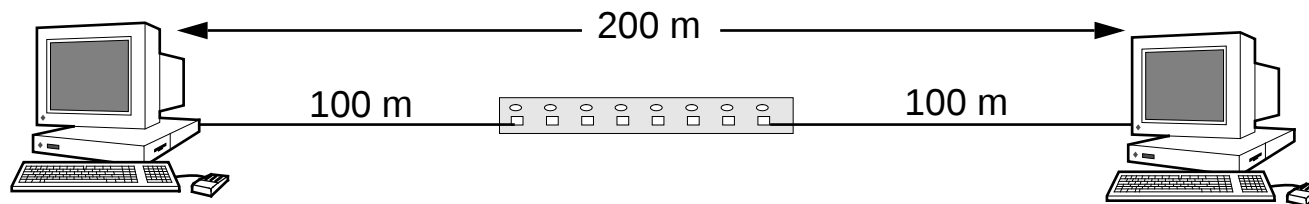
Topologies (cont)



CSMA/CD used in single repeater collision domains
with 1000 Mb/s horizontal runs

Topologies (cont)

- Horizontal runs are 100 meters maximum
- A CSMA/CD topology with horizontal runs out to desktops must support a diameter of at least 200 meters



Bit Budgets

- Bit budget calculations are highly dependent on:
 - Physical layer signalling method and architecture
 - MAC $\Leftrightarrow$ PHY data path width
 - MAC state machine frequency
 - Repeater data path width
 - Repeater state machine frequency
 - Fairness issues
- See Stephen Haddock's presentation for an updated bit budget analysis

Bit Budgets (cont)

- **Conclusions from bit budget analysis**
 - Cable delay dominates, but even with short cables, the bit budget exceeds 512 BT
 - DTE and repeater delay estimates may be optimistic
 - The minimum frame size must be increased from 512 bits to achieve useful topologies at 1000 Mb/s
- **Recommend a new minimum frame size of 512 Bytes**
 - Simply change bits to bytes!

Issues

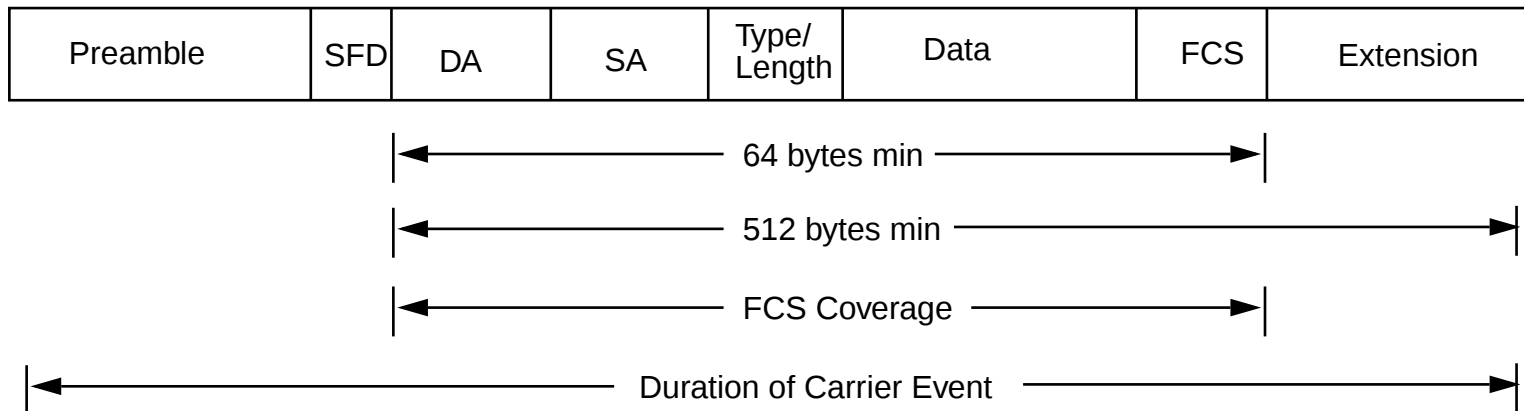
- It's not that simple
- Trying to solve the problem by increasing the minFrameSize to 512 Bytes has harmful side effects
 - Degrades maximum small packet rate on full duplex links where it is not necessary to use the larger minimum size
 - Can't propagate the inflated packets to 100 Mb/s or 10 Mb/s networks, the performance loss would be hideous
 - Can't rely on the padding/stripping mechanism in the 802.3 MAC when using protocol stacks which use the Ethernet frame format
- Must find a way to extend the minimum frame size only on 1000 Mb/s, CSMA/CD networks, regardless of whether Ethernet or 802.3 frame format is used

Solution

- Conceptually, extend the duration of the carrier event without extending the data field, or altering the FCS field
- Introduce a new mechanism which appends non-data symbols to the end of short frames
- Extend the collision window to include these symbols
- Include the extension in the fragment discard calculation
- Remove the extension before checking the FCS and passing the frame to LLC

Solution (cont)

■ Frame extension



- The extension symbols are non-data symbols, and are recognized as such by the MAC and the PHY
- The extension starts on an octet boundary, and is an integer number of octets in length

Impact on MAC

■ New const (4.2.7.1)

const

extendSize = ...; {in bits, implementation-dependent, see 4.4}
extensionBit = ...; {a new type of non-data bit}

■ New Transmit State Variable (4.2.7.2)

var

extension:0..extendSize; {length of extension}

■ New Receive State Variable (4.2.7.3)

var

extendCount: 0..extendSize; {count of extension bits at end of frame}

Impact on MAC

■ Mod to StartTransmit (4.2.8)

```
procedure StartTransmit;  
begin  
    extension := 0;  
    currentTransmitBit := 1;  
    lastTransmitBit := frameSize;  
    transmitSucceeding := true;  
    transmitting := true;  
    lastHeaderBit := headerSize  
end; {StartTransmit}
```

■ Mod to StartReceive (4.2.9)

```
procedure StartReceive;  
begin  
    currentReceiveBit := 1;  
    extendCount := 0;  
    receiving := true  
end; {StartReceive}
```

Impact on MAC (Transmitter)

■ Mod to BitTransmitter (4.2.8)

```
process BitTransmitter;
begin
  cycle {outer loop}
  if transmitting then
    begin {inner loop}
      if halfDuplex then (extension := minFrameSize + extendSize);
      PhysicalSignalEncap; {send preamble and sfd}
      while transmitting do
        begin
          if currentTransmitBit > lastTransmitBit then
            TransmitBit(extensionBit)
          else
            TransmitBit(outgoingFrame[currentTransmitBit]);
          if newCollision then StartJam else NextBit
        end;
      end; {inner loop}
    end; {outer loop}
  end; {BitTransmitter}
```

Impact on MAC (Transmitter)

■ Mod to NextBit (4.2.8)

```
procedure NextBit;  
begin  
    currentTransmitBit := currentTransmitBit + 1;  
    transmitting := ((currentTransmitBit ≤ lastTransmitBit)  
                    or (currentTransmitBit ≤ extension))  
end; {NextBit}
```

■ Mod to StartJam (4.2.8)

```
procedure StartJam;  
begin  
    currentTransmitBit := 1;  
    lastTransmitBit := jamSize;  
    extension := 0;  
    newCollision := false  
end; {StartJam}
```

Impact on MAC (Transmitter)

■ Mod to WatchForCollision (4.2.8)

```
procedure WatchForCollision;  
begin  
  if transmitSucceeding and collisionDetect then  
    begin  
      if currentTransmitBit > (minFrameSize - headerSize + extendSize) then  
        lateCollisionError := true;  
        newCollision := true;  
        transmitSucceeding := false  
      end  
    end {WatchForCollision}
```

Impact on MAC (Receiver)

■ Mod to BitReceiver (4.2.9)

```
process BitReceiver;
  var b: Bit;
begin
  cycle {outer loop}
    while receiving do
      begin {inner loop}
        if currentReceiveBit = 1 then
          PhysicalSignalDecap; {strip off the preamble and sfd}
          b := ReceiveBit; {get next bit from physical Media Access}
          if receiveDataValid then
            if b = extensionBit then
              extendCount := extendCount + 1
            else
              begin {append bit to frame}
                incomingFrame[currentReceiveBit] := b;
                currentReceiveBit := currentReceiveBit + 1
              end
            receiving := receiveDataValid
          end {inner loop}
          frameSize := currentReceiveBit - 1 + extendCount
        end {outerloop}
      end; {BitReceiver}
```

Impact on MAC (Receiver)

■ Mod to ReceiveLinkMgmt (4.2.9)

```
procedure ReceiveLinkMgmt;  
begin  
  repeat  
    StartReceive;  
    while receiving do nothing; {wait for frame to finish arriving}  
    excessBits := frameSize mod 8;  
    frameSize := frameSize - excessBits; {truncate to octet boundary}  
    if halfDuplex then  
      begin  
        receiveSucceeding := ((frameSize ≥ (minFrameSize + extendSize));  
        frameSize := frameSize - extendCount  
      end  
    else  
      receiveSucceeding := (frameSize ≥ minFrameSize);  
    until receiveSucceeding  
  end;
```

MAC Parameters

■ New Parameter Table 4.4.2.4

<u>Parameters</u>	<u>Values</u>
slotTime	4096 bit times
interFrameGap	96 ns
attemptLimit	16
backoffLimit	10
jamSize	32 bits
maxFrameSize	1518 octets
minFrameSize	512 bits (64 octets)
addressSize	48 bits
extendSize	448 octets

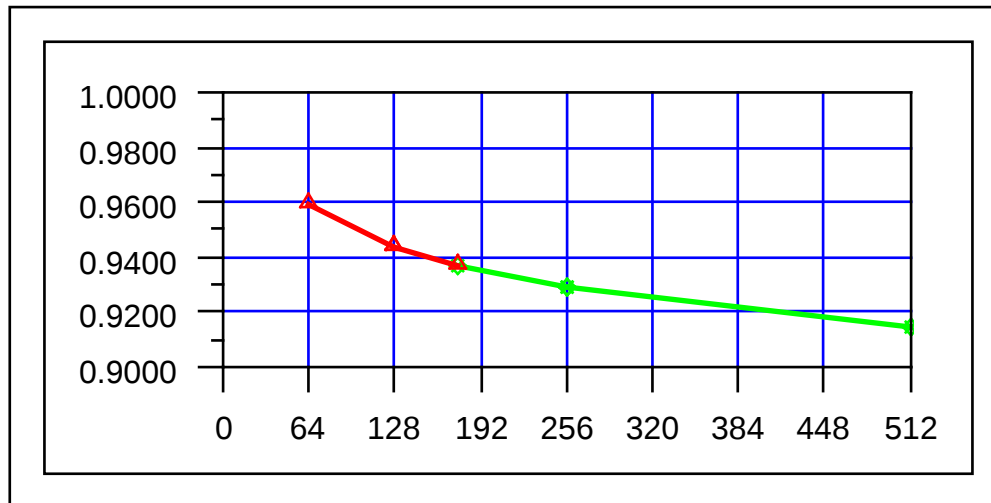
Impact on Reconciliation Sublayer

- **22.2.1.1.2 Semantics of the service primitive PLS_DATA.request**
 - Add EXTEND to the range of values communicated by the OUTPUT_UNIT parameter
- **22.2.1.2.2 Semantics of the service primitive PLS_DATA.indicate**
 - Add EXTEND to the range of values communicated by the INPUT_UNIT parameter

Performance

- Experimental and simulation results presented in La Jolla
(4 node scaled 100BASE-FX using 1500 byte UDP packets)

diameter	200	800	1200	1600	2000
slottime	64	128	174	256	512
txpps	7805.66	7681.15	7626.95	7560.00	7440.00
rxpps	7803.63	7691.94	7634.31	7560.00	7440.00
colps	4835.01	2641.13	1940.92	1420.00	800.00
oerrps	123.45	61.25	46.08	30.00	15.00
udput	0.9592	0.9439	0.9372	0.9289	0.9142
colut	0.0325	0.0313	0.0301	0.0313	0.0340

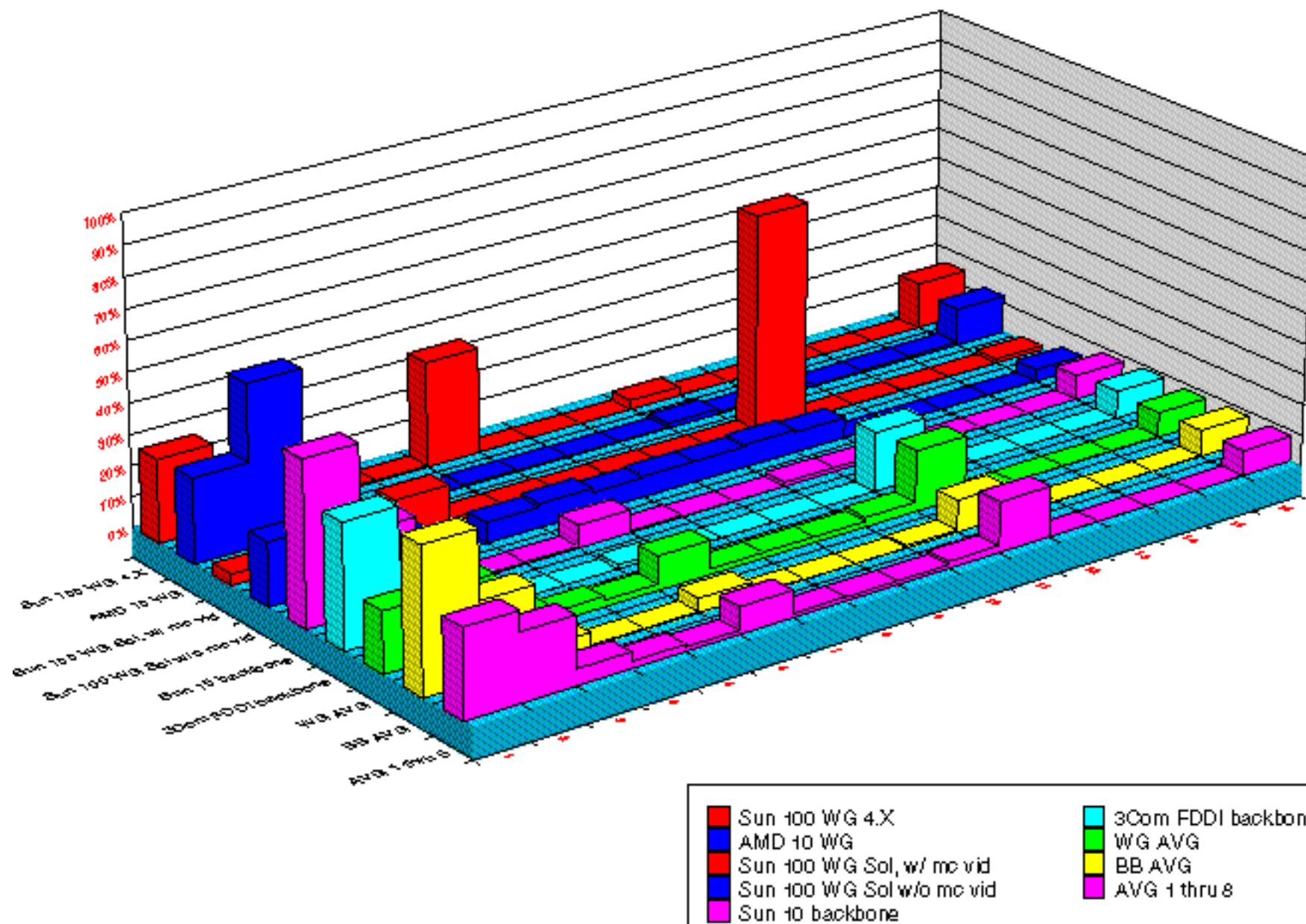


Performance (cont)

- Subsequent simulations used “workgroup average” packet size distribution derived from some “real world” sampling

100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500	1600	Packet Size (less than)
269	.079	.119	.005	.004	.343	.000	.000	.000	.029	.000	.000	.001	.000	.000	.151	1 Sun 100 WG 4.X
275	.527	.012	.005	.003	.006	.004	.002	.003	.016	.005	.002	.004	.008	.009	.123	2 AMD 10 WG
037	.039	.013	.121	.010	.012	.010	.010	.010	.010	.714	.000	.000	.001	.000	.014	3 Sun 100 WG Sol, w/ mc vid
206	.190	.088	.002	.066	.076	.062	.066	.066	.072	.066	.000	.000	.000	.000	.039	4 Sun 100 WG Sol w/o mc vid
530	.252	.018	.004	.005	.074	.005	.002	.001	.013	.012	.001	.001	.001	.001	.081	5 Sun 10 backbone
402	.236	.066	.000	.000	.014	.000	.000	.000	.000	.192	.000	.000	.000	.000	.090	6 3Com FDDI backbone
197	.209	.058	.034	.021	.109	.019	.020	.020	.032	.196	.001	.001	.002	.002	.082	7 WG AVG
466	.244	.042	.002	.003	.044	.003	.001	.001	.007	.102	.001	.001	.001	.001	.086	8 BB AVG
286	.220	.053	.023	.015	.087	.013	.013	.013	.023	.165	.001	.001	.002	.002	.083	9 AVG 1 thru 8

- See Mohan Kalkunte’s presentation from Wakefield for simulation results using “workgroup average” packet size distribution



Conclusions

- Useful CSMA/CD networks can be built at 1000 Mb/s
- The performance of the baseline proposal is adequate for workgroup applications