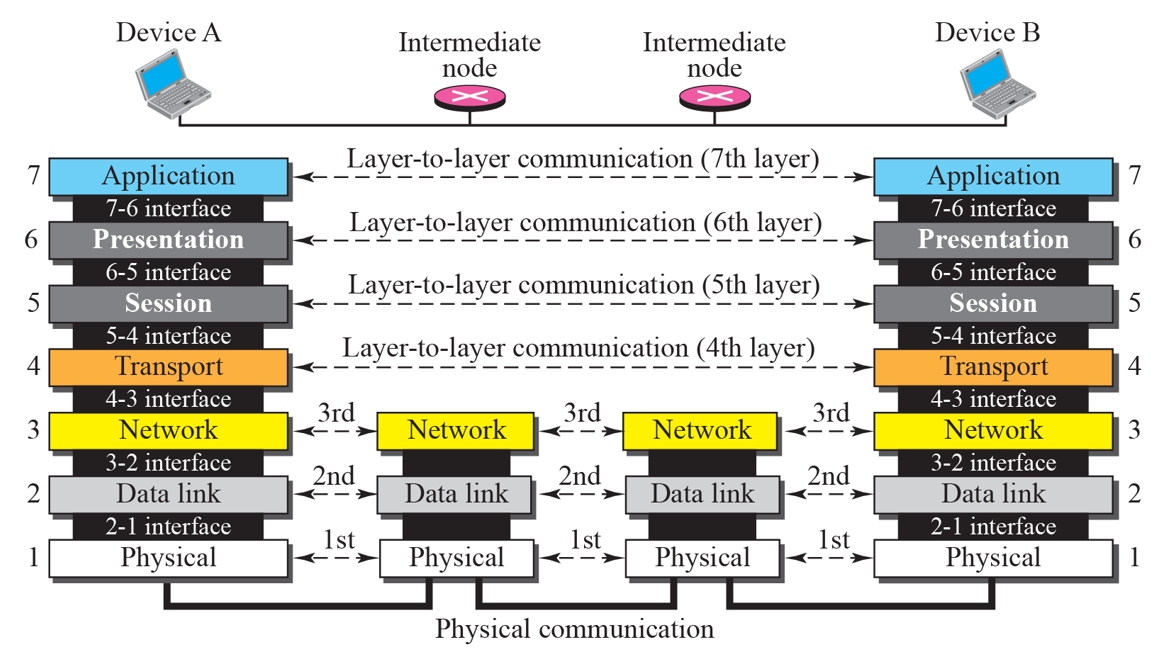
**32776 Midterm Exam (Friday, May 01, 2020) / Reference not described means from MNI Lab**

|  |  |
| --- | --- |
| **01** | Week 01. **Computer Communications Overview [02/26]** |

***01-01: OSI Layers***

|  |  |  |
| --- | --- | --- |
| **7** | Application | **네트워크 자원에 접근**할 수 있게 함 |
| **6** | Presentation | 데이터를 **번역, 암호화, 압축** |
| **5** | Session | **세션을 생성, 관리, 종료** |
| **4** | Transport | 믿을 수 있는 **process-to-process message delivery**와 **error recovery**를 제공 |
| **3** | Network | **패킷을 Source에서 Destination으로 전송**하여 인터넷 제공 |
| **2** | Data link | **Bit를 frame**으로 묶어서 **hop-to-hop delivery**를 제공 |
| **1** | Physical | 개별적인 **bit를 한 노드에서 다른 노드로 이동** |



***01-02. TCP/IP Protocol Suites***

TCP/IP Protocol Suite가 OSI model보다 먼저 개발되었으므로, **TCP/IP Protocol Suite의 각 계층은 OSI model의 각 계층과 일치하지 않는다**.

|  |
| --- |
| TCP/IP 모델과 OSI 모델의 비교 |
| **TCP/IP** Protocol의 **Application** 계층은 **OSI** 모델의 **Application + Presentation + Session** 계층에 해당한다. |

|  |  |
| --- | --- |
| **01** | Week 01. **Computer Communications Overview** |

|  |
| --- |
| TCP/IP Suite의 계층 구조 |
|  |

***01-03. Communication at each layer***

|  |  |
| --- | --- |
| A private internet |  |
| **Physical** layer | * **Physical layer**에서 communication의 단위는 **bit**이다. |
| **Data Link** layer | * **Data Link layer**에서 communication의 단위는 **frame**이다. |
| **Network** layer | * **Network layer**에서 communication의 단위는 **datagram**이다. |

|  |  |
| --- | --- |
| **01** | Week 01. **Computer Communications Overview** |

|  |  |
| --- | --- |
| **Transport** layer | * **Transport layer**에서 communication의 단위는 **segment, user datagram 또는 packet**이다.   + **해당 layer에서 사용되는 프로토콜**에 의해 결정된다. |
| **Application** layer | * **Application layer**에서 communication의 단위는 **message**이다. |

***01-04. Addressing***

TCP/IP 프로토콜에서의 주소 체계:

|  |  |  |
| --- | --- | --- |
| **Message** | **Application** layer | **Application-Specific** addresses |
| **Segment** | **Transport** layer | **Port** addresses |
| **Datagram** | **Network** layer | **Logical** addresses |
| **Frame** | **Data** link layer | **Physical** addresses |
| **Bits** |  |  |

|  |
| --- |
| Example 2.3: 물리적 주소 |
| **1)** 물리 주소 10의 node는 **물리 주소 87의 노드에 frame을 보낸다**.   * Node 10, Node 87은 **LAN이라는 link로 연결**된다.   **2)** Data link layer에서 **프레임의 header에는 물리적(링크) 주소가 포함**된다.   * 이것이 address가 필요한 유일한 부분이다. * Header의 나머지 부분에는 해당 level에서 필요한 다른 정보가 포함된다.   **3)** 물리 주소 **10의 컴퓨터는 sender**이고, 물리 주소 **87의 컴퓨터는 receiver**이다.  **4)** **sender에 있는 data link layer는 상위 계층으로부터 데이터를 수신**한다.   * 이것은 **frame에 있는 데이터를 캡슐화**하고, frame은 **LAN을 통해 전송**된다.   **5)** 물리적 주소가 87이 아닌 각 station은 frame을 **drop**한다.   * **Frame에 있는 Destination address가 고유한 물리적 주소와 다르기** 때문이다. * 원래 의도한 destination에 해당하는 컴퓨터는 **frame에 있는 destination 주소와 물리적 주소의 match**를 찾는다. |

|  |  |
| --- | --- |
| **01** | Week 01. **Computer Communications Overview** |

|  |
| --- |
| Example 2.4: 48-bit(6-byte) physical address |
| 대부분의 local area network는 **12개의 16진수 숫자**로 구성된 **48비트(6바이트) 물리적 주소**를 갖는다.   * 각 바이트(2개의 16진수 숫자)는 **’07:01:02:01:2C:4F’** 와 같이 colon으로 구분된다. |

|  |
| --- |
| Example 2.5: 2개의 라우터가 3개의 LAN을 연결하는 네트워크 |
| **1)** 각 **device(컴퓨터, 라우터)는 각 connection에 대해 주소의 쌍**을 갖는다.   * 여기서는 각 **컴퓨터**가 1개의 링크로 연결되어 있으므로, **1개의 주소 쌍**만을 갖는다. * 각 **라우터**는 3개의 네트워크를 연결하므로, **3개의 주소 쌍**을 갖는다.   **~~2)~~** ~~각 라우터가 서로 다른 물리적 주소를 갖는 것이 당연하겠지만, 왜 각 라우터가 각 connection에 대해 논리적 주소를 필요로 하는지는 불분명하다.~~ (Chapter 11, 12에서 다룸)  **3)** **논리적 주소 A, 물리적 주소 10**의 컴퓨터는 **논리적 주소 P, 물리적 주소 95**인 컴퓨터에 packet을 전송해야 한다. |

|  |
| --- |
| Example 4: IPv4에서의 인터넷 주소 |
| **IPv4**의 인터넷 주소는 **32비트 길이로, 보통 4개의 정수(각 정수는 1바이트)**로 표현된다.   * 각 숫자는 dot으로 구분된다. * Example) 124.63.78.7 |

**(물리적 주소는 각 hop에서 서로 달라질 수 있지만, 논리적 주소는 각 hop에서 서로 같다.)**

|  |  |
| --- | --- |
| **01** | Week 01. **Computer Communications Overview** |

|  |
| --- |
| Example 6: 포트 번호 |
| **Sending computer**는 포트 주소가 각각 **a, b, c인 3개**의 프로세스를 작동시키고, **Receiving computer**는 포트 주소가 각각 **j, k인 2개**의 프로세스를 작동시킨다.   * **Sending computer의 프로세스는 Receiving computer의 프로세스 j와 통신**해야 한다. * 각 컴퓨터는 서로 같은 application을 이용하지만, **하나는 서버이고 하나는 클라이언트이므로 포트 주소는 서로 다를 수 있다**. |

**(물리적 주소는 각 hop에서 서로 달라지지만, 논리적 주소, Port 주소는 보통 서로 같다.)**

|  |
| --- |
| Example 2.7: 16비트 port address |
| **16비트 Port address는 하나의 숫자로** 나타낸다. (예: 753) |

|  |  |
| --- | --- |
| **02** | Week 02. **Data Link Layer Overview [02/27]** |

**[Reference]** <https://terms.naver.com/entry.nhn?docId=3431873&cid=58437&categoryId=58437>

***02-01. Overview***

**데이터 링크 계층(Data Link Layer):** 두 **포인트 간의 신뢰성 있는 전송**을 보장하기 위한 계층

|  |  |
| --- | --- |
| **역할** | 1. 네트워크 위의 **개체들 간 데이터 전달** |
| 1. **물리 계층**에서 발생 가능한 **오류를 찾아내고 수정** |
| 에 필요한 **기능적, 절차적 수단** 제공 |

|  |  |
| --- | --- |
| **02** | Week 02. **Data Link Layer Overview** |

***02-02. MAC address***

**MAC 주소:** **데이터 링크 계층의 상호 통신**을 위한 주소로, 네트워크 카드마다 붙는 **고유한 이름**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **구성:** **총 12개의 16진수 숫자** (예: 00-16-D3-CA-85-67)   |  |  | | --- | --- | | 왼쪽 6개 | **네트워크 카드 제조사, OUI**(Organization Unique Identifier) | | 오른쪽 6개 | **Host Identifier**로, 각 회사에서 임의로 붙이는 Serial 값 | |

***02-03. How Data Link Layer Works***

|  |  |
| --- | --- |
| 스위치 | 데이터 링크 계층의 대표적인 **네트워크 장비** |
| 이더넷 | MAC 계층에서 동작하는 대표적인 **프로토콜** |

|  |  |
| --- | --- |
| 데이터 링크 계층에서 **패킷의 흐름** | 데이터 링크 계층에서의 **OSI계층 패킷 흐름** |
| **물리 계층, 데이터 링크 계층**만 사용하면 **외부 네트워크 없이 스위치만 통과하는 LAN의 통신**과 같이 된다. | 왼쪽의 흐름을 **OSI 7계층 패킷 흐름**으로 나타내면 위와 같다. |
| **<OSI 7계층 기준>**    통신을 위해서는 **패킷이 흘러가기 전에 두 시스템이 서로의 MAC 주소를 알아야** 하므로 스위치도 두 시스템의 MAC 주소를 알아야 한다. | |

**[스위치의 동작 원리]**

|  |
| --- |
| 스위치의 메모리에는 **포트별로 MAC 주소가 매칭된 테이블**이 있고, 그것이 업데이트된다.   * 메모리 테이블에서 **상위 계층인 네트워크 계층의 정보인 IP 주소는 이용하지 않는다**. |

|  |  |
| --- | --- |
| **02** | Week 02. **Data Link Layer Overview** |

**<그림 설명>**

|  |
| --- |
| **[0]** 송신 컴퓨터가 **수신 컴퓨터의 MAC 주소 확인**  **[1]** **패킷의 목적지 MAC 주소** 부분에 **수신 컴퓨터의 MAC 주소**를 입력, **스위치로 전송**  **[2]** 스위치에서 2번 포트로 수신된 **패킷의 목적지 MAC 주소가 3번 포트와 연결된 컴퓨터임을 확인**, 패킷을 **3번 포트로 전송**  **[3]** 패킷이 3번 포트에 연결된 컴퓨터로 전송됨 |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control [02/28, 03/02, 03/03]** |

<https://www.eecs.yorku.ca/course_archive/2010-11/F/3213/CSE3213_11_FlowErrorControl_F2010.pdf>

***03-01. Overview***

|  |
| --- |
| Error Control |
| 1. **Forward Error Correction (FEC)**  * **중복된 정보가 충분히 많을 때** 이것을 이용하여 네트워크상의 오류를 정정  1. **Error Detection + Automatic Retransmission Required (ARQ)**  * Error Correction을 할 수 있을 만큼 **충분한 양의 중복된 정보가 없을 때** 사용  |  |  | | --- | --- | | Receiver detects no errors | **ACK packet이 sender에게 재전송**된다. | | Receiver detects errors | ACK packet이 sender에게 전송되지 않고, **sender는 시간이 지난 후 time-out frame**을 보낸다. | |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

|  |
| --- |
| Challenges of ARQ-based Error Control |
| **Send 1 frame at the time, wait for ACK**   * **구현**하기 쉽지만 **채널 사용에 대해서는 비효율적**이다.   **Send multiple frames at once**   * 채널 사용이 효율적이지만, 구현하기가 어렵다.   + Sender는 **재전송을 위하여 모든 send but unACKed 프레임을 버퍼에 저장**해야 한다. |

|  |
| --- |
| Flow Control |
| **Flow Control:** sender가 **ACK을 기다릴 때 전송할 수 있는 데이터의 양을 제한**하는 데 사용되는 절차의 집합  **<2가지 메인 전략>**   |  |  | | --- | --- | | Stop-and-Wait | Sender는 **ACK을 받을 때까지 기다린 후 다음 프레임을 전송**한다. **(1)** | | Sliding Window | Sender는 **ACK을 받기 전까지 W개의 프레임을 전송**할 수 있다. **(W)** | |

**Error + Flow Control 기술:** **Stop-and-Wait** ARQ, **Go-Back-N** ARQ, **Selective Repeat** ARQ

**Error Detection + ARQ (error detection with retransmissions): outstanding** **(unACKed) 프레임의 개수를 지능적으로 줄이는** 방법을 사용해야 한다.

* **unACKed 프레임의 개수**가 적을수록 **sender와 receiver의 버퍼에 저장된 패킷**이 적다.

***03-02. Stop-and-Wait ARQ***

**Stop-and-Wait ARQ:** 가장 간단한 flow and error control 메커니즘이다.

|  |
| --- |
| 1. Sender는 Receiver에게 **information frame을 전송**한다. 2. 그 다음, Sender는 **stop하고 ACK을 기다린다**. 3. **Time-out**까지 ACK이 도착하지 않으면 Sender는 **frame을 재전송하고, 2로** 간다. |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**Stop-and-Wait ARQ의 비정상적 동작(abnormality) 해결 방법:**

|  |  |
| --- | --- |
| 비정상적 동작 | **Lost acknowledgment** |
| 해결 방법 | **Problem.** Frame은 정상적으로 받았지만 **ACK에서 오류 발생**   * Time-out이 지난 후 **sender가 프레임을 재전송**한다. * 이때 Receiver는 같은 프레임을 2번 받는다.   **Problem. 중복된 프레임의 기각**   * Receiver가 중복된 프레임을 인식할 수 있도록 **프레임에 반드시 번호가 붙여져야** 한다. |

|  |  |
| --- | --- |
| 비정상적 동작 | **Delayed Acknowledgment (Premature Timeout)** |
| 해결 방법 | **Problem.** **ACK이 link 또는 network congestion 문제에 의해 딜레이**될 수 있다.   * **Time-out의 만료 시간을 줄이고**, sender가 frame을 재전송한다. * Delayed ACK이 도착하면 sender는 이것이 **마지막으로 보내지는 프레임에 대한 것이라고 간주**한다.   **Problem.** **Delivered packet의 순서에 대한 gap** 방지   * **ACK에 반드시 번호가 붙여져야** 한다. |

* **Packet, ACK sequence의 크기는 1-bit**이다.

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**<Stop-and-Wait Efficiency>**

|  |
| --- |
| **Basic Stop-and-Wait delay (t0):** **프레임이 채널로 전송**된 시각부터 **ACK이 receiver에게 도착하고 다른 프레임이 전송**되는 시각 사이의 간격  **Effective Transmission (data) rate ():**    **Transmission Efficiency (): 실제 transmission rate와 Effective transmission rate**의 비율   * **:** **bandwidth-delay product**   + 각 시점에서 전송되는 bit의 개수의 최댓값   + **Stop-and-Wait ARQ**에서는 delay-bandwidth product는 **전송되는 비트 수에 대한 lost opportunity의 정도**이다.   + Sender->Receiver과 그 back에서의 **전송 파이프의 용량**   Stop-and-Wait ARQ는 데이터가 여러 개의 조각으로 나눠져 있을 때, **즉 이 에 비해 작을 때 부적합**하다. |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**<Stop-and-Wait ARQ example>**

|  |
| --- |
| * Stop-and-Wait는 **속도가 매우 빠르거나 propagation delay가 큰 경우** 잘 작동하지 않는다. |

**<Stop-and-Wait Efficiency in Channel with Errors>**

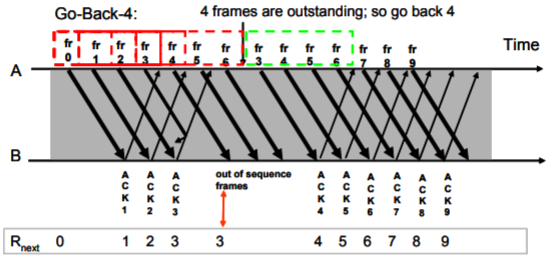
|  |
| --- |
| 가 전송된 프레임에 오류가 있어서 재전송되어야 할 확률일 때,   * **Total delay per frame:** * **가 증가할수록 는 감소**한다.      * **초항, 공비가 모두 인 무한등비급수의 합**   **<Total average delay per frame>** |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**<Piggybacking>**

|  |
| --- |
| Stop-and-Wait ARQ는 **단뱡향 통신**이고, **양방향 통신에서는 각 party가 데이터를 send & acknowledge**한다.  **Piggybacking method:** *outstanding ACK는 information 프레임의 맨 앞에 위치한다.*   * **데이터 프레임과 ACK 프레임으로부터의 오버헤드는 하나의 프레임**으로 합쳐질 수 있으므로 **bandwidth를 절약**한다. |

***03-03. Go-Back-N ARQ***

**Go-Back-N ARQ:** Sender는 **ACK을 기다리는 동안 채널을 busy하게 유지할 수 있을 만큼 많은 양의 프레임**을 보내서 **Stop-and-Wait ARQ의 비효율성을 극복**한다.

* **Outstanding frame**이 허용되는 window인 를 사용한다.
* **m비트 순서 값**이 **frame과 ACK 양쪽**에 모두 사용된다.

|  |
| --- |
| 라고 하면,   1. sender는 프레임을 하나씩 보낸다. 2. **frame 3은 전송 오류**가 발생하고, **receiver는 frame 3과 이후의 프레임을 무시**한다. 3. sender는 결국 **outstanding frame의 최대 개수**에 도달한다. 4. Sender는  **프레임만큼** 돌아가고, **frame 3 이후의 모든 frame을 재전송**한다. |

**<Sender Sliding Window>**

|  |
| --- |
| **모든 프레임은 버퍼에 저장**되어 있다. (단, **outstanding frame은 window 내부**에 있음)   * Window의 **left 이전의 frame**은 이미 **ACK되어서 purge**되었다. * Window의 **right 이후의 frame**은 **window가 slide될 때까지 전송될 수 없다.** * **새로운 ACK 도착** 시 window는 새로운 전송되지 않은 **frame을 포함하도록 slide**된다. * Window가 outstanding frame의 최대 개수에 도달하면 모든 window는 resent된다. |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**<Receiver Sliding Window>**

|  |
| --- |
| Receiver window의 **크기는 항상 1**이다.   * Receiver는 항상 **특정한 frame이 특정한 order에 도착하기**를 기다린다. * **Out of order에 도착하는 frame**은 모두 기각되며 재전송되어야 한다. |

**“Go-Back-N의 receiver의 Complexity는 항상 Stop-and-Wait과 같다.”**

**<Problem with Go-Back-N: Go-Back-N with Timeout>**

|  |
| --- |
| Go-Back-N 알고리즘은 sender에게 무한한 패킷이 공급될 때 잘 작동된다. 그러나 **패킷이 산발적으로 공급될 때** 문제가 발생할 수 있다.   * 그 다음에 개의 패킷을 전송할 수 없으면 **window는 exhaust되지 않고, 재전송도 되지 않을** 것이다. * 이것은 **Go-Back-N을 다음과 같이 수정하여 해결**할 수 있다.   + 1. 각각의 전송되는 **프레임에 대해 타이머**를 설정한다.   + 2. **Window가 가득 차거나 첫 번째 frame의 타이머가 만료**되었을 때 모든 outstanding frame을 재전송한다. |

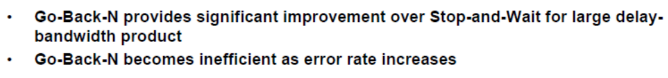
|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

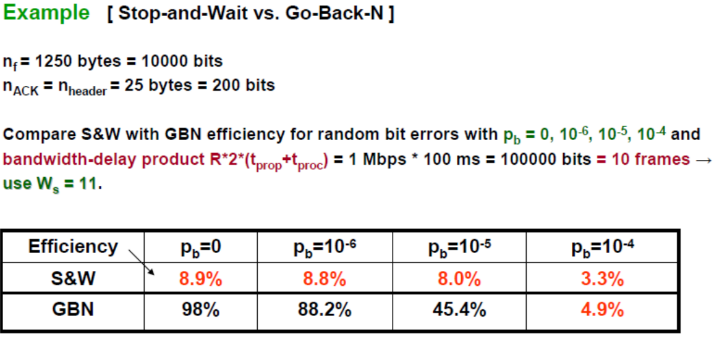
**<Sequence Numbers and Window Size>**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 헤더에 sequence number에 대한 비트로 m비트가 할당되었을 때, 가능한 sequence number의 가짓수는 이다.   |  |  | | --- | --- | | Sender window size |  | |  | **여러 개의 frame이 같은 seq. number**를 가지므로 **ambiguous ACK**이 발생하고, 따라서 **accept될 수 없다**. | |  | 다음과 같이 ambiguity가 발생할 수 있다. | |  | Accept될 수 있다. | |

**<Go-Back-N Efficiency>**

|  |
| --- |
| * 의 값이 **channel을 busy하게 유지할 수 있을 만큼 크면서 channel에서 오류가 발생하지 않으면** 최적의 효율성을 가진다. * Error-prone channel에서 이면 **frame delivery time**은 다음과 같다.   **<Total avg. time required to transmit a frame>**  **<Transmission efficiency>** |

**<Example>**



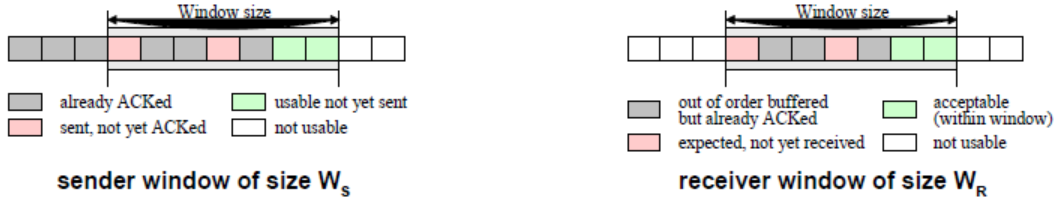
|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

***03-04. Selective Repeat ARQ***

**Selective Repeat ARQ:** 다음의 2가지 특징을 추가하여 Go-Back-N의 한계점을 극복한다.

|  |  |
| --- | --- |
| Go-Back-N의 한계점 | **Not suitable for ‘noisy links’**   * **Lost/damaged frame의 전체 window가 resent**되어야 한다. * 과도한 재전송은 bandwidth를 다 쓰게 하여 전송을 느리게 만든다. |
| Receiver window>1 frame | **Out-of-order이지만 error-free한 프레임**이 accept될 수 있게 한다. |
| 재전송 메커니즘 수정 | **개별적인 프레임**만 재전송된다. |

* Selective Repeat ARQ는 **TCP에서 사용**된다.



**<Selective Repeat ARQ Opreation>**

|  |  |
| --- | --- |
|  | Receiver |
| * Window는 **다음 in-order frame이 도착**하는 대로 증가한다. * **Out-of-order** 프레임은 **seq. number**가 다음 조건을 만족시킬 때만 accept된다. * **Out-of-order frame이 발견**되는 대로 Sequence number **의 Negative ACK (NAK)**이 전송된다. |
| Sender |
| * Window는 **ACK**이 도착하는 대로 증가한다. * **Timer가 만료**되면 대응되는 **frame이 resent되고 timer는 리셋**된다. * **NAK이 도착**하는 대로  **frame이 resent**된다. |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

**<Window Size and >**

|  |
| --- |
| Header에 sequence number로 m비트가 할당된 경우 **가능한 sequence number는 가지**   * **은 아래와 같이 ambiguity**가 생길 수 있으므로 적합하지 않다. * **이 적합**하다. |

**<Selective Repeat Efficiency>**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 가 **channel을 busy하게 유지할 만큼 크면서 channel이 error-free하면 최적의 효율성**을 갖는다.   * 단, **sequence number space는 Go-Back-N의 2배**가 되어야 한다. * Error-prone channel에서는…   **<Average time required to transmit a frame>**     |  |  | | --- | --- | | 1번째 시도 성공 |  | | 2번째 시도 성공 |  | | 평균 |  | |

|  |  |
| --- | --- |
| **03** | **Week 03. Error and Flow Control** |

***03-05. Performance Comparison: Stop-and-Wait vs. Go-Back-N vs. Selective Repeat***

**<과 가 에 비하여 무시할 만큼 작을 때>**

|  |
| --- |
|  |

**<3가지 ARQ 테크닉의 효율성>**

|  |
| --- |
|  |

* 일 때 Selective Repeat의 성능이 가장 좋다.
* 일 때 Go-Back-N의 성능은 Selective Repeat의 성능에 가까워진다.

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol [03/04]** |

<https://www3.nd.edu/~cpoellab/teaching/cse40815/Chapter6.pdf>

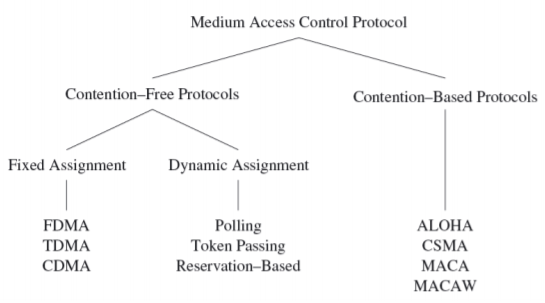
***04-01. Overview***

**Medium Access Control (MAC) Protocol:** 여러 개의 node가 **통신 매체를 공유할 때, 이 매체에 대한 접근을 제어**하는 프로토콜

* MAC protocol의 선택 및 설계는 **네트워크 통신의 신뢰성 및 효율성, 에너지 효율성**과 관련되어 있다.

**<MAC layer의 역할>**

|  |
| --- |
| 1. **Node가 언제 공유된 매체에 접근**할지 결정 2. 경쟁하는 node 간의 잠재적인 **충돌 해결** 3. **물리 계층**에서 발생하는 **통신 오류 정정** 4. **Framing, addressing, flow control** 등을 수행 |

**<MAC Protocol의 분류>**

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

***04-02. Contention-Free Medium Access***

**Contention-Free Medium Access:** 각 node가 **자신에게 할당된 자원을 배타적으로 사용**하여 충돌을 방지할 수 있다.

**[Fixed assignment Strategies]**

|  |  |
| --- | --- |
| **비효율적:** **모든 프레임에서** **필요하지 않다면 한 device에 할당된 slot을 다른 device로 재할당하는 것이 불가능**하기 때문 | |
| **Frequency** Division Multiple Access  **(FDMA)** | Frequency band는 **여러 개의 작은 frequency band**로 나뉜다.   * 한 쌍의 노드 간에 **데이터가 이동할 때 하나의 frequency band**를 사용한다. * **모든 다른 노드는 서로 다른 frequency band**를 사용한다. |
| **Time** Division Multiple Access  **(TDMA)** | **여러 개의 device가 서로 같은 frequency band**를 사용한다.   * **주기적인 time window (frame)**에 의존한다.   + Frame은 **서로 다른 device에서의 매체 접근을 분산**시키기 위한 정해진 개수의 전송 슬롯을 갖는다.   + **Time schedule:** 어떤 노드가 특정 슬롯에서 데이터를 전송할 것인가? |
| **Code** Division Multiple Access  **(CDMA)** | **Code를 통해 Wireless medium에 동시 접속**하는 것을 허용한다.   * code들이 **orthogonal**하면 **같은 frequency band를 공유**하여 multiple communication이 가능하다. * **Receiver에서의 Forward Error Correction (FEC)**는 **동시 통신에서의 간섭을 복구**하는 데 사용된다. |

**[Dynamic assignment Strategies]**

|  |  |
| --- | --- |
| Polling-based protocols | Controller device가 **polling frame을 발급하여 round-robin 방법으로 각 station에 할당**한다. 이때 각 station에 전송할 데이터가 있는지 묻는다.   * 전송할 데이터가 없으면 다음 station에 묻는다. |
| Token passing | Station은 **다른 station에 token이라는 특별한 frame을 이용하여 polling request를 보낸다.** (round-robin 방법)   * Station은 **token을 가지고 있을 때만 데이터를 전송**할 수 있다. |
| Reservation-based protocols | 잠재적인 매체 접근을 **예약하기 위한 static time slot**이 사용된다.   * 각 node는 정해진 위치에 있는 reservation bit에 togging을 하여 전송 의사를 전달할 수 있다. |

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

***04-03. Contention-Based Medium Access***

**Contention-Based Medium Access:** 각 node는 **동시에 전송에 착수**할 수 있다.

* **충돌 횟수를 줄이고 충돌로부터 복구**할 수 있는 메커니즘이 필요하다.

**<ALOHA 계열 프로토콜>**

|  |  |
| --- | --- |
| ALOHA protocol | **Broadcast 데이터전송이 성공했는지 확인하기 위한 acknowledgement**를 사용한다.   * **Node가 매체에 즉시 접근**하는 것을 허용한다. * 성공적인 전송 가능성을 높이기 위해 **Exponential back-off와 같은 접근 방법으로 충돌을 해결**한다. |
| Slotted-ALOHA protocol | Station은 **미리 정해진 시점(time slot의 시작점)에만 전송을 시작**할 수 있다.   * ALOHA의 효율성을 증가시킨다. * **각 노드 간의 동기화**가 필요하다. |

**<CSMA 계열 프로토콜> - Carrier Sense Multiple Access**

|  |  |
| --- | --- |
| CSMA with Collision Detection **(CSMA/CD)** | Sender는 먼저 **매체가 busy/idle한지 판단**한다.   * busy이면 패킷을 전송하지 않고, idle이면 전송할 수 있다. |
| CSMA with Collision Avoidance **(CSMA/CA)** | CSMA/CD에서 sender가 충돌이 발생할지 판단해야 한다면, CSMA/CA에서는 **먼저 충돌을 피하려고 시도**한다. |

**<Hidden and Exposed Terminal Problems>**

|  |
| --- |
| **Hidden-terminal problem:**   * sender A와 sender C는 모두 B에 도달할 수 있지만, **서로의 signal을 수신할 수 없다.** * **A, C는 모두 B에 데이터를 보낼 수 있으며,** 충돌을 직접 감지할 수 없으면 B에 충돌을 일으킨다.   **Exposed-terminal problem:**   * C는 D에 데이터를 보내려고 하지만, **B에서 A로 전송하는 것을 탐지하여 대기한다**. * 이때 **B의 전송은 C에서의 데이터 수신에 간섭할 수 없다**. |

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

**<CSMA 계열 프로토콜에 대한 TMI>**

Node는 **전송을 시작하기 전에 먼저 medium을 감지**하여 충돌 횟수를 줄인다.

|  |  |
| --- | --- |
| **Non-persistent** CSMA | 노드는 **medium이 idle일 때 한번 즉시 데이터를 전송**할 수 있다.   * Medium이 **busy 상태이면 back-off 연산**을 하고, **재전송하기 전에 특정 시간만큼 기다린다**. |
| **1-persistent** CSMA | 노드는 계속 데이터를 전송하려고 하지만 **medium이 busy 상태인 것을 계속 감지**한다.   * Medium이 **한번 idle 상태가 되면 즉시 데이터를 전송**한다. * **충돌** 발생 시 재전송 전에 **random period of time만큼 기다린다**. |
| **p-persistent** CSMA | 노드는 **medium을 계속 감지**한다.  **Random back-off value**는 다음의 둘 중 하나이다. |
| CSMA**/CA**  (CSMA with Collision Avoidance) | 노드는 medium을 감지하지만, **idle로 판단되었을 때도 channel에 즉시 접근하지 않는다**.   * 대신, **DCF interframe space (DIFS) + (slot size의 배수)** 만큼의 time period 동안 대기한다. * Medium에 접근하려는 node가 여러 개이면 **back-off period가 짧은 node**가 접근하게 된다.   **[예시]**  Node A는 **DIFS+4\*s** 동안 기다리고 Node B의 back-off는 **DIFS+7\*s**이다. (s: slot size)   * Node A가 전송을 시작할 때 Node B는 **자신의 back-off timer를 정지**하고 **A가 전송을 완료한 후 DIFS만큼 지났을 때 timer를 재개**한다. * Node B의 back-off timer가 종료되면 Node B도 전송을 시작할 수 있다. |

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

**<MACA 계열 프로토콜> - Multiple Access with Collision Avoidance**

|  |  |
| --- | --- |
| Multiple Access with Collision Avoidance  **(MACA)** | **Dynamic한 예약 메커니즘**을 갖는다.   * Sender는 **ready-to-send (RTS) 패킷**을 통해 전송 의사를 표시한다. * Intended Receiver는 **clear-to-send (CTS) 패킷**을 통해 응답한다.   + Sender가 CTS를 받지 않으면 다음 시점에 재전송을 시도한다. * RTS와 CTS를 **overhear하는 node는 예약이 이미 되었으므로 기다려야** 한다. (데이터 전송 크기 등에 기반하여 결정) * **Hidden terminal problem을 해결하여 충돌 횟수를 줄인다.** |
| MACA for Wireless LANs  **(MACAW)** | **Receiver는 데이터 수신 후 ACK을 통해 응답**한다.   * Receiver의 범위 내에 있는 **다른 node들은 channel이 available하다는 것을 인식**한다.   **RTS를 hear하지만 CTS를 hear하지 않는** node는 전송이 발생할 것인지를 알지 못한다.   * MACAW는 **data sending (DS) 패킷**을 이용하는데, 이것은 성공적인 handshake를 알리기 위해, **CTS를 받은 후에 sender에 의해 전송**된다. |

**MACA-BI (MACA by Invitation):** destination device는 **Ready To Receive (RTR) 패킷을 Source에 전송**하여 데이터 전송을 개시한다.

* **MACA와 비교하여 오버헤드가 적은데**, 그 이유는 다음과 같다.

|  |
| --- |
| 1. 이론상의 **최대 throughput이 증가**한다. 2. Destination이 **데이터를 언제 수신할지를 아는지에 의존**한다. |

* Source node는 **queued message의 수를 나타내는 optional field**를 데이터 메시지에 추가하여 사용할 수 있다. (**더 많은 RTS 패킷이 필요하다는 정보**를 destination에 제공)

***04-04. IEEE 802.11***

**IEEE 802.11:** IEEE(Institute of Electrical and Electronics Engineers)에 의해 1999년에 발표된, **무선 연결을 위한 OSI 모델의 물리 계층과 데이터 링크 계층을 명시**하는 프로토콜

* **Wi-Fi (Wireless Fidelity)**라고 부르기도 한다.
* **CSMA/CA와 MACAW의 개념을 결합**하였으며, **에너지를 보존**하기 위한 특성도 제공한다.

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

**<IEEE 802.11의 연산 모드>**

|  |  |
| --- | --- |
| **Point** Coordination Function (PCF) | **Access point (AP) 또는 Base Station (BS)**라고 하는 central entity를 중심으로 하는 device 간의 통신 **(managed mode)** |
| **Distributed** Coordination Function (DCF) | 각 device는 서로 직접 통신한다. **(ad-hoc mode)** |

**<CSMA/CA를 기반으로 한 IEEE 802.11>**

|  |
| --- |
| Node가 데이터를 전송하기 전에 **medium의 상태를 확인**한다.   * Medium의 상태가 **최소 DCF interframe space (DIFS) 동안 idle**일 때 데이터를 전송할 수 있다. * 그렇지 않으면 device는 전송을 연기하기 위해 **back-off 알고리즘**을 수행한다.   + 이 알고리즘은 몇 개의 **time slot을 랜덤하게 선택**하고, 이 **time slot의 값을 back-off counter에 저장**한다.     - 네트워크에서의 활동 없이 지나가는 **모든 time slot에 대해 counter의 값이 감소**한다.     - Counter의 값이 **0이 되면 device는 전송을 시도**한다.     - Counter의 값이 **0이 되기 전에 네트워크 활동이 감지**되면, device는 counter의 값을 **감소시키기 전에 channel이 DIFS의 기간 동안 idle일 때까지 대기**한다.   **성공적으로 전송**이 이루어진 후에,   * Receiver device는 **SIFS (short interframe space)**만큼 기다린 후에 acknowledgment를 통해 응답한다.   + **SIFS < DIFS인 이유:** 어떤 다른 device도 **receiver가 acknowledgement를 보내기 전에 접근하지 못하게** 하기 위해서   Node A가 **RTS, DTS control message를 통해 reservation**을 하면,   * RTS message를 overhear하는 **이웃한 node B**는 A의 전송이 끝나고 **acknowledge될 때까지 매체에 접근하지 말아야** 한다.   + 이것은 B가 **medium이 다시 idle**이 되는지 확인하기 위해 **지속적으로 medium의 상태를 감지**해야 한다는 것을 의미한다. |
| **NEXT PAGE** |

|  |  |
| --- | --- |
| **04** | **Week 04~06. MAC Protocol** |

**<CSMA/CA를 기반으로 한 IEEE 802.11> (cont.)**

|  |
| --- |
| **PREVIOUS PAGE** |
| 대신, A의 RTS message에 **전송할 데이터의 크기가 포함**되어 있으면,   * Node B가 데이터를 전송하는 데 **시간이 얼마나 걸릴지 예측**하고 **low-power sleep mode에 진입할지 결정**하게 한다. * 이웃한 노드는 **CTS를 overhear할 수 있지만 RTS를 overhear할 수는 없으므로**, 데이터 크기는 **CTS message에 저장**된다. * 데이터 크기 정보를 이용하여 이웃한 노드가 매체가 **얼마나 오랫동안 unavailable할지를 나타내는 NAV (network allocation vector)**를 설정할 수 있다.   + 이것은 매체를 계속 감지할 필요성을 줄여서 **node가 전력을 절약**하게 한다. |

**[PCF mode]**

|  |
| --- |
| Access point (AP)는 **충돌로부터 자유로운 통신을 보장**하기 위하여 **채널 접근을 대등**하게 한다.   * **주기적으로 beacon에서 client device들로 브로드캐스팅**하고, 이때 AP에서 보류 중인 데이터가 있는 device의 목록을 포함한다. * **Contention-free period** 동안 AP는 client device에 이 패킷들을 전송한다. * AP는 client device들이 데이터 전송을 개시할 수 있도록 투표한다. * AP는 **PCF interframe space (PIFS) 라는 wait period**를 이용한다.   + **SIFS < PIFS < DIFS**   + CTS, ACK과 같은 **DCF mode에서의 control message와의 간섭 없이**, PCF 트래픽이 **DCF mode에서 작동하는 device**들에 의해 생성되는 트래픽보다 **우선 순위가 높다**는 것이 보장된다. |

* IEEE 802.11은 **높은 throughput과 mobility를 지원**하면서 **매체에 대한 공평한 접근**을 제공한다.
  + Device가 **medium을 감지하는 데 오랜 시간**을 사용하고 **충돌이 자주 발생**하면 오버헤드가 커지고, 따라서 에너지가 많이 소비된다.
* IEEE 802.11은 **PCF mode**에서 작동하는 기기에 **PSM (power saving mode)**을 제공한다.

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview [03/05]** |

***05-01. Overview***

**Internet as a…**

|  |  |
| --- | --- |
| Block Box |  |
| Combination of LANs and WANs connected together |  |

***05-02. Switching***

|  |  |
| --- | --- |
| **Circuit** Switching | 모든 메시지는 **패킷으로 분리되지 않고** Source에서 Destination으로 전송된다.   * **예시 – early telephone systems, telephone network:** callee가 응답하면 circuit이 생성되고, 모든 연결된 device가 circuit을 유지하는 동안 voice message가 양방향으로 흐를 수 있다. |
| **Packet** Switching | 메시지는 **Source에서 manageable한 패킷으로 분리**된 다음에 전송되고, **그 패킷들은 destination에서 assemble**된다.   * **네트워크 계층**은 packet-switched network로 설계되어 있다.   **<네트워크 계층의 Packet-switched network>**   |  | | --- | | Source에 있는 패킷은 **datagram이라는 manageable한 패킷으로 분리**된다.   * Datagram은 Source에서 destination으로 전송된다. * 수신된 datagram은 destination에서 원래 메시지를 생성하기 전에 assemble된다.   **인터넷의 Packet-switched 네트워크 계층은 원래 connectionless service로 설계**되었으나, 최근에는 **connection-oriented service로 바꾸려는 경향**이 있다. | |

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview** |

***05-03. Connectionless Service and Connection-Oriented Service***

**<CONNECTIONLESS packet-switched network>**

|  |  |
| --- | --- |
| **Connectionless packet-switched network** |  |
| **Forwarding process** in a connectionless network | **Forwarding은 packet의 destination address에 의해 결정**된다. |
| **Delay** in a  connectionless network |  |

**<CONNECTION-ORIENTED packet-switched network>**

|  |  |
| --- | --- |
| **Connection-oriented packet-switched network** | Connectionless network에서와 달리 **모든 패킷이 virtual circuit을 따라 차례대로 이동**한다. |

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview** |

|  |  |
| --- | --- |
| **Forwarding process** in a connection-oriented network | **Forwarding은 packet의 label에 의해 결정**된다. |
| **Virtual circuit** | **<virtual-circuit network에서의 request packet 전송>**  R1에서 **Outgoing port가 3**이므로 3을 따라 **R3**으로 이동한다.  R3에서 **Outgoing port가 3**이므로 3을 따라 **R4**로 이동한다.  R4에서 **Outgoing port가 4**이므로 4를 따라 **B**로 이동  **<virtual-circuit network에서의 acknowledgement setup>**  request packet의 반대 방향으로 이동한다. 즉 이번에는 **Incoming Port의 값에 따라 이동**한다.  **<virtual-circuit에서의 패킷의 흐름>**  모든 패킷이 **request packet과 ACK setup의 경로를 따라 이동**한다. |

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview** |

|  |  |
| --- | --- |
| **Delay** in a  Connection-oriented network | Connectionless network에서 **Setup time, Teardown time이 추가**되었다. |

***05-04. Network Layer Services***

**<Source Computer에 의해 제공되는 서비스>**

|  |
| --- |
| 1. Data+Len+DA+PI+ST는 **datagram으로 패킷화**된다. 2. **Destination의 논리적 주소**를 이용하여, **다음 hop의 논리적 주소**를 Routing Table에서 찾아서 datagram에 추가한다. 3. **다음 hop의 논리적 주소**를 이용하여, **다음 hop의 MAC 주소**를 찾아서 추가한다. 4. **Datagram+MAC이 fragment된다**. 이때 link를 MTU table에서 찾아서 MTU 값을 찾는다. |

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview** |

**<각 라우터에서의 프로세싱>**

|  |
| --- |
| 1. Datagram이 **valid datagram이면 수신하고, 그렇지 않으면 기각**한다. 2. **<Source Computer에 의해 제공되는 서비스>**의 2~4와 같다. |

**<Destination Computer에서의 프로세싱>**

|  |
| --- |
| 1. 데이터 링크 계층으로부터 **datagram을 받아서 valid datagram이면 수신하고, 그렇지 않으면 기각**한다. 2. **Depacktetize**한 후 **각 fragment의 데이터를 저장**한다. 3. **모든 fragment가 도착하면 reassemble**한다. 그렇지 않으면 **reassembly timer가 만료된 경우 모든 fragment를 기각하고, 그렇지 않으면 대기**한다. 4. **reassemble된 데이터**를 상위 계층으로 전송한다. |

|  |  |
| --- | --- |
| **05** | **Week 07. Network Layer Overview** |

***05-05. Error and Flow Control***

**<데이터 링크 계층에서의 오류 검사>**

|  |
| --- |
| * 현재 버전의 **인터넷 네트워크 계층에서는 flow control(흐름 제어)이 제공되지 않는다**. |