Network

بسم الله والحمد لله والصلاة والسلام على رسول الله

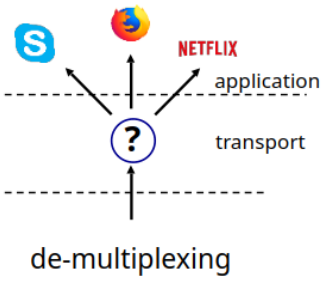
تنبيه: هذا فقط ملخص للفكرة العامة لا التفاصيل

Transport layer services

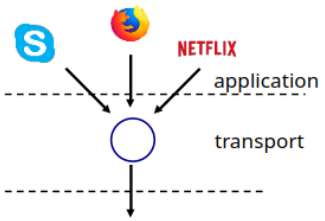
- ال(transport layer) توفر اتصالا أو (logical communication) بين ال(processes) ليس كال(network layer) توفر اتصالا بين الأجهزة نفسها (hosts)
- هناك طريقتان لفعل هذا باستخدام (transport layer)
  - UDP: ارسال الرسالة بأقل مساحة ممكنة بدون التأكد من وصولها ولا أي شيء ..... فقط أرسل وانتظر الأفضل

    هذا يسمى (best effort approach)
  - TCP: طريقة ارسال محكمة لها عدة ميزات مثل التالي:
    - reliable, in-order delivery
    - congestion control
    - flow control
    - connection setup
  - لكن ليس هناك طريقة معينة في ال(transport layer) لفعل التالي:
    - delay guarantees (تحديد الوقت المطلوب للاتصال قبل فعله)
    - bandwidth guarantees (تحديد السرعة والمساحة المتاحة قبل الإرسال)

Multiplexing and demultiplexing



de-multiplexing

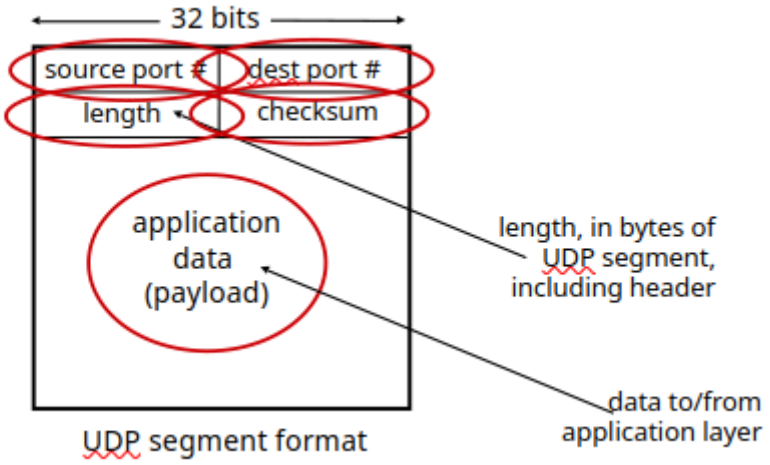


multiplexing

عملية ال(multiplexing) تتم عند ال(sender) بحيث تمنح ال(packet) معلومات في ال(header) تساعد في عملية ال(demultiplexing) عند ال(receiver)

- Multiplexing, demultiplexing: based on segment, datagram header field values
- UDP: demultiplexing using destination port number (only)
- TCP: demultiplexing using 4-tuple: source and destination IP addresses, and port numbers
- Multiplexing/demultiplexing happen at all layers

Connectionless transport: UDP



لاحظ احتواء ال(header) على أربع خانات منها ال(checksum) وهو عبارة عن (ones complement) لمحتوى ال(UDP segment) ... في بعض الحالات يمكن أن يكون خاطئا !

- "no frills" protocol:
  - segments may be lost, delivered out of order
  - best effort service: "send and hope for the best"
- UDP has its plusses:
  - no setup/handshaking needed (no RTT incurred)
  - can function when network service is compromised
  - helps with reliability (checksum)
- build additional functionality on top of UDP in application layer (e.g., HTTP/3) -> إذا أردت استخدام هذا النوع لكن تريد المزيد من الميزات يمكنك إضافة الميزات في الطبقة الأعلى (application)

Principles of reliable data transfer

باختصار المطلوب في هذه النقطة فهم بعض الأشياء التي ستسهل علينا فهم ال(TCP) فيما بعد ... هنا كأننا نبني (protocol) جديد لإرسال البيانات في ال(transport layer) ... نريد أن نجعله يتعامل مع مشكلتين مهمتين: (packet loss and corruption) ...

هذا ال(protocol) الذي سنؤسسه لنسميه (reliable data transfer protocol) أو (RDT) اختصارا وسنستخدم في فهمه رسمة تسمى (FSM: finite state machines)..وفي ما يلي سأوضح ال(versions) المختلفة له والمشاكل التي سيحلها وكيف ...

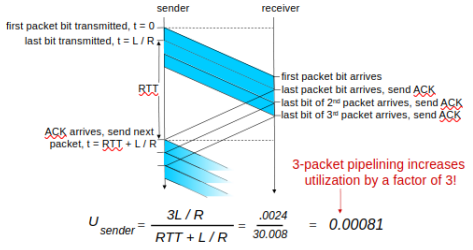
1. افترض أن RDT 1.0 يتعامل كأنه لا يوجد خطر (packet loss or delay) في ظروف مثالية غير موجودة في الحياة الواقعية ..
2. في RDT 2.0 سنحل مشكلة ال(bit error: packet corruption) بإضافة checksum و ACK and NACK في ال(receiver) لكي يعلم ال(sender) هل العملية تمت بنجاح أم لم تتم وإذا لم تتم بنجاح يعيد إرسال الرسالة من جديد وسيتمتع ال(protocol) الآن بخاصية (stop and wait)

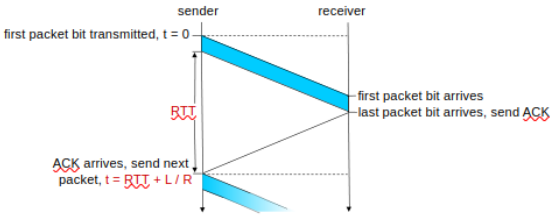
3. في RDT 2.1 سنحل مشكلة ال(duplicates and corrupt ACK/NAK) باستخدام (sequence number for every packet) هذا سيجعلنا نميز كل رسالة على حدة
4. في RDT 2.2 سنستخدم فقط ACK ولن نستخدم NAK لأنه يمكننا الاستغناء عنها ... إذا حدثت مشكلة في أحد ال(packets) عند ال(receiver) لا يرسل NAK إنما يرسل ACK عليها ال(sequence number) لآخر (packet) صحيحة فقط حينها يعرف ال(sender) أن هناك خطأ بعد هذه ال(packet) الصحيحة ويرسل ما بعدها من ال(packets) مرة أخرى
5. في RDT 3.0 سنحل مشكلة (packet loss) باستخدام (countdown timer) يعيد إرسال الرسالة إذا لم يرسل ال(receiver) رسالة ACK عليها في الوقت المحدد

- يمكننا تحسين ال(performance) بتحسين ما يسمى (utilization) وهذا قانونه:

$$U_{sender} = \frac{L / R}{RTT + L / R}$$

- وهو الوقت المستغرق في إرسال الرسالة مقارنة بالوقت الكلي
- هذا التحسين لل(utilization) سيتم باستخدام طريقة (pipelining) وهذا يتطلب
  - range of sequence numbers must be increased
  - buffering at sender and/or receiver
  - cumulative ACK: ACK(n): ACKs all packets up to, including seq # n
- هذا الفرق بين استخدام (pipelining) وعدم استخدامه:





- لل(pipelining) طريقتان هما
  - Go-back-N: uses cumulative ACK .. and can buffer or not the received out-of-order messages

  - selective repeat: ACKs individual packets and have a timer for each one also ... it buffers the packets sent out-of-order