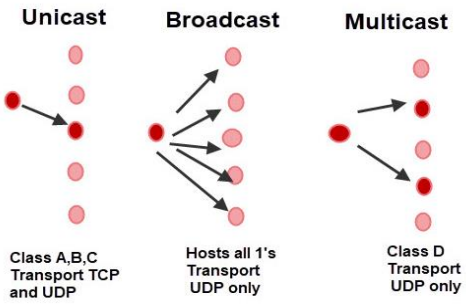
INFO SHEET

Multicast Communication

Uni-, Broad, Multicast



- **Unicast**: Punkt zu Punkt- Übertragung. Die Daten werden von einem Endpunkt zu einem anderen Endpunkt übertragen.
- **Broadcast**: Rundsendung: Die Daten werden von einem Endpunkt an alle Endpunkte in einer Broadcast-Domain verteilt. Der Empfänger muss dann entscheiden ob er die erhaltenen Daten verarbeiten möchte oder nicht.
- **Multicast**: Punkt zu Mehrpunkt-Übertragung. Die Daten werden von einem Endpunkt gesendet und von den Knotenpunkten an diejenigen Empfänger verteilt, welche die Daten angefordert haben

Basic-Multicast

- sichert zu, im Gegensatz zu IP Multicast, dass ein korrekter Prozess die Nachricht schließlich ausliefert, solange der Multicaster nicht abstürzt
- Eine verlässliche Unicast-Send-Operation (one-to-one send operation) wird für die Implementierung verwendet:
  - To B-multicast(g,m): for each process p ϵ g, send(p,m);
  - On receive*(m)* at p: *B-deliver(m)* at *p*.
- **Nachteil:**
- „ack-implosion": Die Acknowledgements, die gesendet werden können von vielen Prozessen gleichzeitig ankommen, sodass sich der Puffer des Multicasting-Prozesses schnell füllt.
  - Gefahr, dass diese verloren gehen
  - Auslastung der Netzwerkbandbreite
- **Lösung:**
- praktische Implementierung von Basic Multicast kann mithilfe von IP-Multicast umgesetzt werden
- mithilfe von Sequenznummern köbnnen fehlende Nachrichten erkannt und angefordert werden, und es entsteht keine ack-Implosion.

Reliable multicast over IP multicast

1. Protokoll geht davon aus, dass die Gruppen geschlossen sind. Es verwendet:
   - Piggybacked acknowledgements (ack die an anderen Nachrichten angehängt sind)
   - Negative acknowledgements (wenn Nachrichten versäumt werden)

IP- Multicast

- Implementierung einer **Gruppenkommunikation**
- ermöglicht dem Sender IP Pakete **an viele Empfänger, die eine Multicast-Gruppe bildet** zur gleichen Zeit zu senden ohne die Identität der einzelnen Empfänger und die Größe der Gruppe zu kennen
- Eine Multicast-Gruppe wird durch eine **Internetadresse der Klasse D** spezifiziert, d.h. eine Adresse, deren erste 4 Bits in 1110 in IPv4 (Adressbereich:224.0.0.0 – 239.255.255.255) sind
- Die Mitgliedschaft in Multicast-Gruppen ist **dynamisch**, jeder Computer kann jederzeit ein- oder austreten und einer beliebigen Anzahl an Gruppen beitreten & es ist möglich Datagramme über eine Multicast-gruppe zu senden ohne Mitglied zu sein
- wird durchgeführt indem es **UDP-Datagramme** mit Multicast Adressen und Port-Nummern sendet
- durch das erstellen eines Sockets können Nachrichten der Gruppe empfangen werden

- **Failure model:**
  - IP-Multicast arbeitet UDP-basiert und somit ist die Nachrichtenzustellung nicht garantiert, da UDP bestätigte, verbindungslose Kommunikation verwendet d.h.
  - **Ommision failures**: Einige aber nicht alle Mitglieder können eine Nachricht erhalten
  - **Unrelibale multicast**: IP-Pakete kommen nicht in Absenderreihenfolge an & Gruppenmitglieder können Nachrichten in unterschiedlicher Reihenfolge empfangen

Reliable Multicast

- ***Integrity:*** Ein korrekter Prozess p liefert eine Nachricht m höchstens einmal aus (at-most-once-delivery). Darüber hinaus ist *p ϵ group(m)* und m wurde von *sender(m)* durch den Multicast-Vorgang verschickt.
- ***Validity:*** Wenn ein korrekter Prozess die Nachricht *m* per multicast versendet oder empfängt, dann wird *m* garantier ausgeliefert
- ***Agreement:*** Wenn ein korrekter Prozess Nachricht *m* ausliefert, dann wird *m* auch allen anderen korrekten Prozesse in *group(m)* zugestellt

Ordered Multicast

- **FIFO-ordering:**
- Wenn ein korrekter Prozess zunächst *multicast(g, m)* und dann *multicast(g, m')* ausführt, dann liefert jeder korrekte Prozess aus g, der *m'* ausliefert, *m* vor *m'* aus
- lokale Senderreihenfolge wird garantiert
- **Causal-ordering:**
- Wenn multicast(g, m) → multicast(g, m') mit → als „happens-before"-Operator als logische Reihenfolge gilt, dann liefert jeder korrekte Prozess aus g, der m' ausliefert, m vor m' aus
- garantier die Zustellung nach der Reihenfolge, die durch die Relation „→" festgelegt wird
- **Total-ordering:**
- Wenn ein korrekter Prozess eine Nachricht m vor einer Nachricht m' ausliefert, dann liefert jeder korrekte Prozess aus g, der m' ausliefert, m vor m' aus.
- garantiert die Empfangsreihenfolge über alle Prozesse der Gruppe

2. Jeder Prozess *p* verwaltet :
   - Sequenznummer *S(p,g)* für jede *group(g)* zu der er gehört
   - Sequenznummer *R(q,g)*, der letzten Nachricht, die er vom Prozess *q* geliefert bekommen hat
3. Wenn p eine Nachricht *m R-multicastet* dann:
   - piggyback S(p,g) und gebe acknowledgements für empfangene Nachrichten in folgender Form <q,R(q,g)>
   - IP multicast die Nachricht an g, und erhöhe die Sequenznummer S(p,g) um 1
4. Vorgehen beim Empfang einer Nachricht von q mit Sequenznummer S gilt:
   - Wenn S=R(p,g) +1: R-deliver die Nachricht und erhöht R(p,g) um 1
   - Wenn S<= R(p,g): verwerfe die Nachricht, da diese schon empfangen wurde
   - Wenn S> R(p,g)+1 oder wenn R>R(q,g), for enclosed acknowledgement<q,R>:
   - dann wurde Nachricht verpasst und fordere mit einer negative acknowledge an
   - stelle neue Nachrichten in die Warteschlange für eine spätere Zustellung